Sistemas Informáticos
Curso 2002-03

Simulación de nuevas arquitecturas en Internet

Saulo Barajas Fernández
Santiago Martínez Sanz
Jaime Parodi Bardón

Dirigido por:
Prof. Javier García Villalba
Dpto. de Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

Índice

RESUMEN

El elevado aumento de usuarios de Internet que se ha producido en los últimos años está exigiendo cada vez más servicios a una Red que no fue pensada para soportarlos. Ya no es suficiente con aplicaciones como el correo electrónico o las páginas Web, sino que los usuarios comienzan a utilizar Internet para aplicaciones en tiempo real como videoconferencia o telefonía.

Sin embargo, la actual Internet no soporta una adecuada calidad de servicio, esto es, no es capaz de distinguir los paquetes que deberían servirse con mayor prioridad. El presente trabajo estudia los distintos parámetros que influyen en la calidad de servicio, tales como el retardo y la pérdida de paquetes, y propone diversos escenarios de simulación tanto con calidad de servicio como sin ella. Se generan distintos tráficos a través de una topología de red y se analizan los distintos parámetros de la calidad de servicio mediante gráficas. Las simulaciones se realizan con la herramienta de simulación de redes, ns. Finalmente se extraen conclusiones de los distintos resultados obtenidos.

SUMMARY

The huge growth of Internet users during the last years demands new services for a network no thought and designed to support them. Applications like electronic mail or web pages aren't enough, the users begin to use Internet for real time applications like videoconferences or telephonic applications.

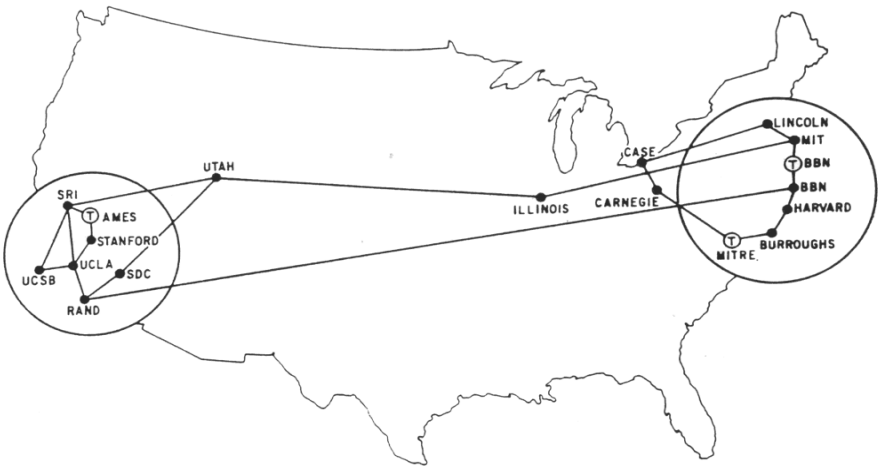
However, current Internet doesn't support a suitable quality of service, the net is enable to difference the packets that would be served with priority. The present research studies different parameters that influence over quality of service like delay, packet loss and proposes several situations with and without quality of service. Different traffic types are generated over a net topology and the quality of service parameters are analyzed by means of graphics. Simulations are developed with the net simulator toolkit, ns. Finally, conclusions are obtained by different results.

PALABRAS CLAVE

Redes, Internet, calidad de servicio, QoS, ns, simulación, retardo, paquetes, rendimiento.

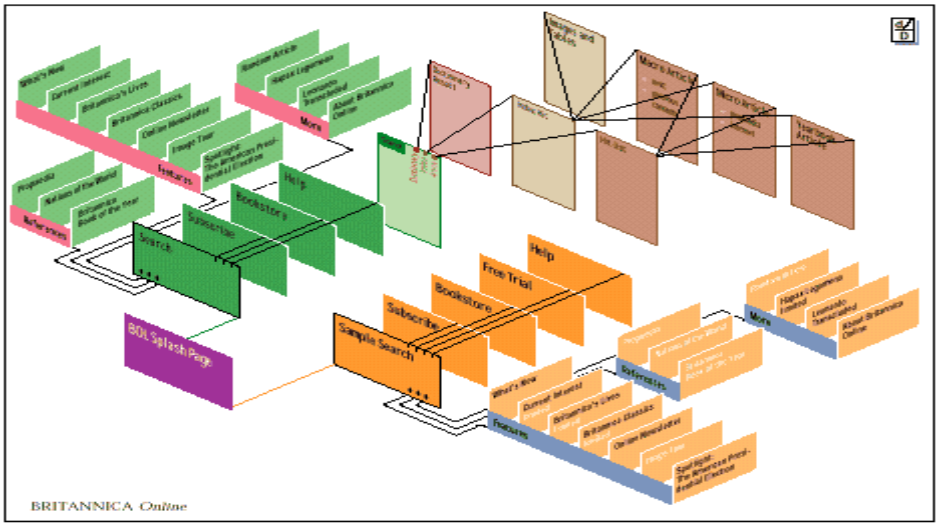
1. INTRODUCCIÓN

Las primeras ideas de lo que es Internet nacen en los años 60 tanto por una necesidad militar para comunicarse de una manera fiable y segura como por la necesidad de la comunidad científica para compartir investigaciones, de esta manera surgió Arpanet. Esta red primitiva será la precursora de la actual Internet.



Nodos de la red Arpanet a principios de los años 70

Lo que en principio fue diseñado para una utilización limitada y restringida sufre un gran cambio a finales de los años 80 y principios de los 90 con el paso de esta red para la utilización de forma masiva y para usos comerciales. Este hecho es provocado por el uso extendido del protocolo *http* y el nacimiento del World Wide Web.

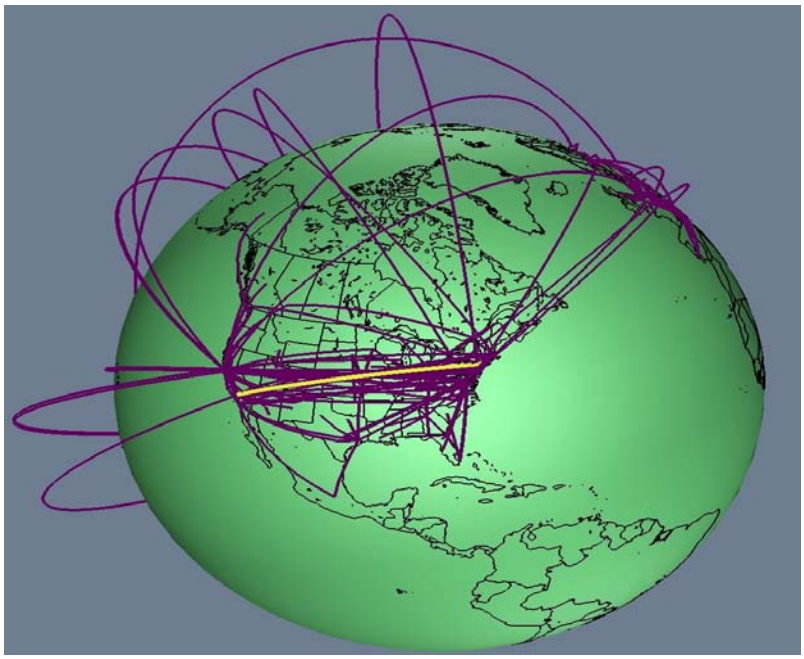


Mapa del Web Site de Britannica Online

Las aplicaciones de Internet están soportadas por el conjunto de protocolos TCP/IP que es el estándar a nivel mundial para la interconexión de sistemas abiertos. Ningún otro protocolo ofrece tanta interoperabilidad o abarca tantos fabricantes de sistemas. TCP/IP está compuesta de 2 capas:

- Protocolo IP: es el responsable de mover los paquetes de datos de un nodo a otro. IP permite el envío de un paquete a una dirección formada de 4 bytes, que es conocida como dirección IP. Las autoridades de Internet son las encargadas de asignar estas direcciones IP a los distintos organismos o miembros solicitantes de las mismas.
- Protocolo TCP: es el responsable de verificar el correcto reparto de los datos desde un cliente al servidor. Los datos se pueden perder en la red intermedia y por tanto es necesario un control de errores para que la comunicación sea efectiva.

Hoy en día, esta evolución no ha dejado de continuar, convirtiéndose Internet en medio y herramienta necesarios en todos los sectores tanto laborales como de ocio. Esta evolución implica expansión a nivel de usuarios y a nivel de cantidad de información a comunicar. Pero la evolución tiene un límite y la tarea de transmitir información multimedia y de tiempo real a través de la red es dónde se encuentran grandes retos, es decir, no ofrecen una calidad de servicio adecuada. Para solucionar estos problemas no basta con modificar el protocolo de Internet IP, ni de crear otro nuevo, si no lo que es necesario es una nueva Arquitectura de Red.



2. OBJETIVOS DEL PROYECTO

Una vez descrito el concepto de Internet a grandes rasgos y comentado someramente el problema de la calidad de servicio (QoS) que proporciona la red para la transmisión de datos asociadas a aplicaciones de tiempo real, pasamos a describir el objetivo del proyecto.

Antes de dar solución a un problema de tal magnitud como sería el diseño y desarrollo de un nuevo sistema, en este caso, el diseño de una nueva Arquitectura de Red, es necesario un proceso de simulación. Este proceso de simulación nos ayuda a comprender por una parte el actual estado de Internet, mediante el descubrimiento de carencias, faltas y disfunciones y por otra podemos probar un sistema diseñado antes de desarrollarlo.

La labor del proyecto es realizar una serie de simulaciones de distintos escenarios de ejemplos de tráficos que pueden circular a través de la Red, midiendo y comparando los parámetros que indican la calidad de servicio que proporciona una red.

EL simulador empleado para realizar el proyecto es el *Network Simulator* conocido como *ns*, del que más adelante hablaremos más profundamente.

3. CALIDAD DE SERVICIO

La calidad de servicio también conocida como QoS (Quality of Service) es la capacidad de la red para proporcionar un mejor servicio sobre un tráfico seleccionado.

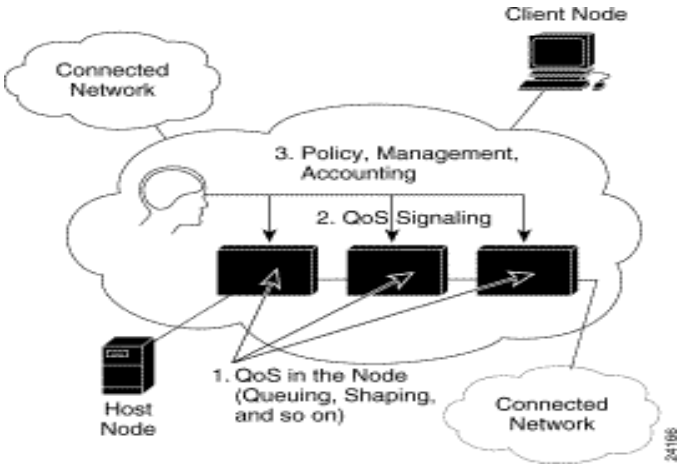
El principal objetivo de la calidad de servicio es proporcionar por una parte cierta prioridad incluyendo ancho de banda dedicado, variación de retardo controlada y latencia y por otra mejorar las características de pérdida. Es importante asegurar que proporcionando prioridad para uno o más flujos no signifique que el resto de flujos fallen.

Por lo tanto, las características de la calidad de Servicio son:

- Asignación de ancho de banda.
- Evitando y/o administrando la congestión de la red.
- Manejo de prioridades a través de la red.
- Modelación del tráfico de la red.

Las funciones y mecanismos necesarios para proporcionar una adecuada calidad de servicio dependerán de los siguientes aspectos:

- Políticas para administración para el control del tráfico de un extremo a otro a través de la red.
- Colas, planificación (Scheduling) y características del tráfico.
- Selección del tráfico.
- Técnicas de señalización para coordinar QoS de principio a fin entre los elementos de la red.
- Administración y control de la congestión.



3.1. PARÁMETROS PARA MEDIR LA CALIDAD DE SERVICIO

Para medir la calidad de servicio de una red de paquetes se utilizan distintos parámetros o medidas como son los siguientes: rendimiento o throughput, retardo o delay, variación del retardo o jitter, pérdida de paquetes, goodput y badput.

Esta serie de parámetros se irá describiendo a continuación.

3.1.1 Rendimiento o Throughput

El valor del rendimiento o throughput nos dice la cantidad de información que la red es capaz de transportar durante un intervalo de tiempo de extremo a extremo. Más concretamente es la cantidad de tráfico recibido y medido en destino deseado. En una red óptima el valor del rendimiento debería coincidir con la capacidad ideal de la red, pero en una red real sólo un porcentaje del tráfico ofrecido a la red es recibido satisfactoriamente en su destino.

3.1.2. Retardo o Delay

Es conocido que en toda la red de comunicaciones se produce un retardo. Este retardo, que es el tiempo empleado en transmitirse el tráfico entre la aplicación de origen y destino, presenta dos componentes: un primero, que es el retardo generado por los nodos de la red en el procesamiento de los paquetes (análisis de la cabecera IP y de las tablas de enrutamiento, así como en la clasificación y planificación de los paquetes), y un segundo que es el retardo debido a la propagación en los medios de transporte: fibra óptica, enlace por satélite, etc. Una gráfica típica se muestra en la figura 2, donde se observa que a mayor rendimiento de la red mayor es el retardo promedio experimentado por los paquetes.

3.1.3. Variación del Retardo o Jitter

La variación del retardo de los paquetes en una red o jitter ocurre principalmente cuando los paquetes de un mismo flujo experimentan retardos diferentes cuando atraviesan los nodos de la red. El principal efecto de un valor significativo de la variación del retardo es la presencia de ráfagas de tráfico durante la transmisión de una aplicación. Si bien es cierto que este parámetro puede ser un factor de degradación para aplicaciones multimedia, el hecho de que dependa directamente del retardo de la red hace que su medida no sea un aspecto indispensable. Aún así, se puede calcular como la diferencia entre el retardo de un paquete y el retardo promedio de los paquetes en la red.

3.1.4. Relación de pérdida de paquetes o Packet Loss Ratio

Entre la causa más significativa de que se produzcan pérdidas de paquetes en una red está la congestión. Esto sucede cuando los buffers de los nodos sufren un desbordamiento o cuando se está en los límites del retardo permitido. La relación de pérdida de paquetes nos da una idea de la calidad de la conexión. El efecto directo de las pérdidas de paquetes R está relacionado con la calidad de la aplicación en el receptor y se define como el cociente entre el número de paquetes perdidos y el número de paquetes transmitidos. Las aplicaciones intolerantes pueden soportar hasta un valor de referencia para R para que sean consideradas con una adecuada QoS.

3.1.5. Goodput y Badput

Este parámetro está relacionado con los datos recibidos satisfactoriamente por un receptor (throughput) que fluyen a través de una red de paquetes. De los paquetes recibidos algunos serán de utilidad y otros no. La razón de los datos útiles se denomina goodput y la razón de los datos sin ninguna utilidad se denomina badput. El origen de estos dos parámetros se debe a que protocolos de nivel superior pueden sufrir fragmentación, y aunque algunos de estos fragmentos lleguen a su destino, otros pueden ser descartados por algún router en la red. En el momento de ensamblar los fragmentos recibidos en el receptor pertenecientes al protocolo de nivel superior, algunos de éstos pueden ser inútiles, descartando de esta manera a los paquetes (en este caso a los fragmentos) recibidos satisfactoriamente en el receptor. En general, el rendimiento o throughput es igual a la suma del goodput con el badput.

4. CARENCIAS DE IPv4 PARA PROPORCIONAR QoS

La actual versión del Protocolo de Internet (IP) es la versión Ipv4. Este protocolo viene siendo usado desde los años 70, desde el nacimiento de lo que hoy llamamos Internet.

Las principales características de esta versión del Protocolo de Internet a grandes rasgos son las siguientes:

- Utiliza direcciones de 4 bytes (32 bits) que identifican cada terminal comunicante. Un ejemplo de dirección sería: 128.24.45.98.
- La localización de direcciones es ineficiente y restrictiva.
- Las direcciones de Ipv4 son difíciles de obtener.
- El diseño de este protocolo no soporta los actuales requerimientos de la red.
- El número de usuarios máximo de Internet se puede llegar a quedar corto, 2^{32}=4294967296, es el máximo número de usuarios debido a que las direcciones son de 32 bits.

Estas limitaciones de la red no fueron previstas como dejan constancia las citas de personajes históricos de la informática, mostradas a continuación:

"*I think there is a world market for maybe five computers*".
Thomas Watson, IBM, 1943.

"*640 K should be enough for anybody".*
Bill Gates, 1981.

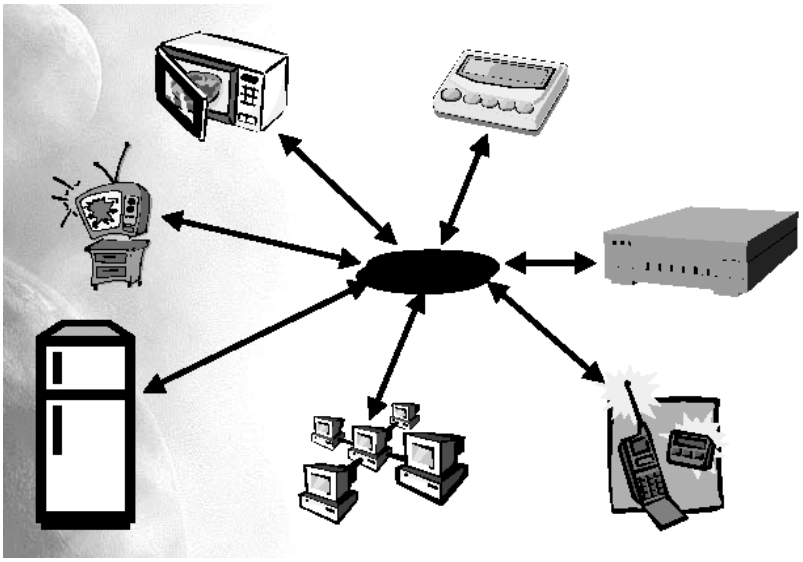
"*32 bits ought to be enough address space*".
Vint Cerf, 1977.

El protocolo Ipv4 está presente en el mundo de una manera dominante, debido al crecimiento de dispositivos con direcciones IP, tales como terminales móviles, electrodomésticos... y por tanto al crecimiento de usuarios que utilizan Internet.

Este es el gran problema de este protocolo, las expectativas de uso que fueron definidas a la hora del diseño han sido superadas y serán aún más. El uso que actualmente se pretende para Internet pasa por la comunicación de ciertas aplicaciones en tiempo real y multimedia como son los campos del cine, música, videojuegos en los sectores de ocio, videoconferencias en los sectores profesionales que requieren una calidad de servicio que actualmente no es proporcionada.

5. PROPUESTAS DE IPv6 PARA MEJORAR LA QoS

IPv6 es el protocolo que debe sustituir a IPv4 para proporcionar esa calidad de servicio no conseguida. Para ello este protocolo no debe constar de ciertas mejoras sino que debe nacer de la idea de un cambio de Arquitectura de Red.



Internet se va a convertir en el punto de unión de numerosos dispositivos y por tanto debe tener capacidad de comunicación suficiente para que ese gran número de dispositivos puedan convivir y realizar sus distintos cometidos sin estorbarse unos a otros.

Los beneficios del protocolo IPv6 se pueden resumir en los siguientes puntos:

- Incremento del espacio de direcciones. Las direcciones pasan a estar formadas por 128 bits permitiendo un muchísimo mayor número de usuarios y dispositivos que pueden formar parte de la red.
- Encaminamiento más eficiente. Esto es debido que la asignación de direcciones asociadas es desde el principio, las cabeceras de los paquetes IP poseen una longitud fija y que la cabecera ofrece capacidad de expansión.

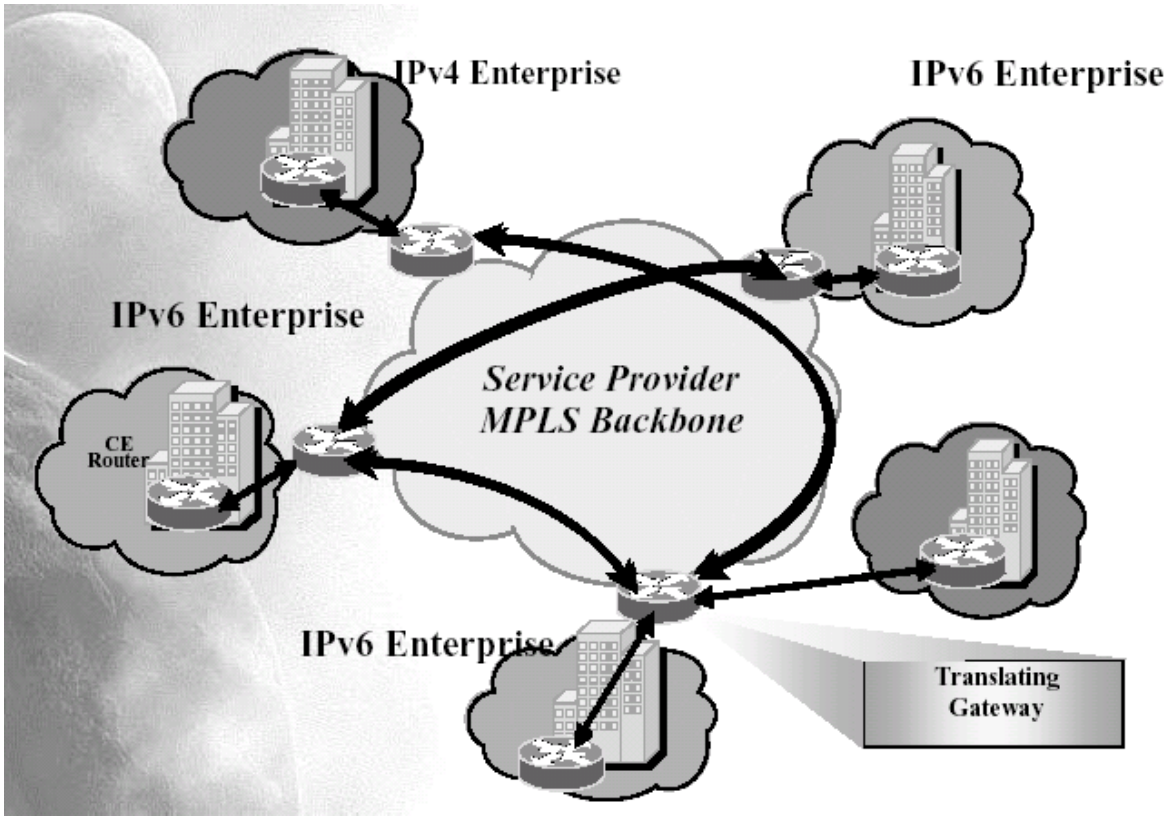
Estos beneficios son los productores de esa calidad de servicio deseada.

La principal diferencia de la cabecera de los paquetes aparte de que posee una longitud fija es que consta de dos campos relacionados con la calidad de servicio:

- Etiqueta de flujo, este campo de 20 bits está asociado a la IntServ, también conocida como la Internet de Servicios Integrados.
- Indicador de clase de tráfico, tiene una longitud de 8 bits y está asociado a DiffServ, también conocida como la Internet de Servicios Diferenciados.

Cómo hemos dicho anteriormente, un hecho fundamental es que IPv6 no está concebido como una mejora de IPv4, sino que el concepto de este protocolo es el de una nueva Arquitectura de Red, es decir, IPv6 se sustenta sobre la arquitectura MPLS.

Esta nueva Arquitectura de Red surge de la necesidad de distintos requerimientos y exigencias por parte de los usuarios y dispositivos que se van a conectar. Estos requerimientos pueden ser tales como la convergencia de voz y datos, la necesidad de unos servicios de extremo a extremo en tiempo real, agrupación de telefonía y vídeo. A su vez esta nueva Arquitectura facilita la movilidad y el acceso, con lo que hace estas labores más fáciles a los teléfonos y equipos móviles, hoy en día el estandarte de las comunicaciones.



6. MIGRACIÓN DE IPv4 a IPv6

El cambio o migración de IPv4 a IPv6 es una tarea difícil. Esta dificultad radica en el enorme tamaño de Internet y la gran cantidad de usuarios que tiene. Muchas organizaciones de gran entidad están siendo día a día más dependientes de la actual Internet y por tanto, no se muestran a favor del cambio. La migración debe hacerse nodo a nodo con procedimientos de auto-configuración para evitar la necesidad de configurar manualmente los hosts que albergarán IPv6.

IPv4 presenta algunas características que simplificarían la migración, ejemplos de estas características son el tipo de direcciones, las direcciones de IPv6 se derivarían de las de IPv4. Otro ejemplo sería que los túneles con los que se implementa IPv6 pueden ser construidos sobre las redes de IPv4 o al menos en una fase inicial.

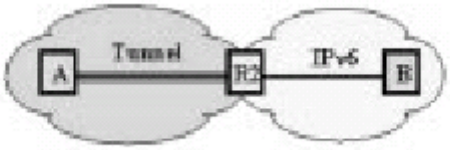
Los objetivos principales de la migración se pueden resumir en los siguientes puntos:

- IPv4 e IPv6 deben ínteroperar.
- El uso de clientes y conmutadores deben estar distribuidos en Internet como un camino simple y progresivo.
- Los administradores de red y usuarios finales deben pensar que la migración es fácil de entender e implementar.
- Posibilidad de una transición progresiva y no traumática, los nodos de IPv4 deben ser actualizados a nodos de IPv6 sin requerir que otros nodos se actualicen automáticamente.
- Requerimientos mínimos para la actualización.
- Direccionamiento simplificado.
- Coste inicial bajo, sin necesidad de un trabajo preparatorio.
- Una estructura de direcciones IPv6 que permite la derivación de direcciones IPv6 desde IPv4.
- La disponibilidad de una pila dual durante la transición que signifique la convivencia de ambas pilas, la de IPv4 e IPv6.
- Una técnica para encapsular los paquetes de IPv6 dentro de paquetes de IPv4.
- Una técnica opcional que permita traducir las cabeceras de IPv4 a cabeceras de IPv6 y viceversa.

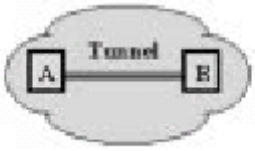
Mientras la infraestructura de encaminamiento para IPv6 está siendo desarrollada, el encaminamiento estará basado en el de IPv4. El *tunneling* es un conjunto de técnicas que permiten transportar tráfico de IPv6 a través de las redes de IPv4. Durante la migración, el *tunneling* puede ser usado de de diferentes modos o métodos:

Conmutador a Conmutador

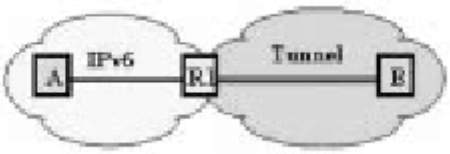
Nodo a Conmutador



Nodo a Nodo



Conmutador a Nodo



7. ALGORITMOS DE CONTROL DE ADMISIÓN

El control de admisión decide si se admite o no un determinado flujo que quiere ingresar en la red teniendo en cuenta su estado actual. Es un proceso que regula el tráfico de la red para limitar el número de flujos admitidos en la red de tal manera que cada flujo individual obtenga la QoS deseada. Cuando se realiza el proceso de admisión de un nuevo flujo, el algoritmo de control de admisión no solamente decide si el flujo puede acceder a los servicios requeridos, sino que también decide si la admisión del flujo no va a alterar los compromisos adquiridos previamente por la red con respecto a otros flujos.

7.1. Tipos de Control de Admisión

Existen dos aproximaciones al control de admisión: *control de admisión basado en parámetros* y *control de admisión basado en medidas*. El control de admisión basado en parámetros se basa en las características previas o a priori del tráfico en términos de los parámetros de un modelo determinístico o estocástico, por lo que la decisión de admisión es el resultado de un análisis matemático detallado. En esta aproximación se busca el caso peor en el análisis estadístico del tráfico de un nuevo flujo y su impacto sobre los flujos existentes. Esta aproximación puede ser usada para soportar QoS garantizada o estadística. El control de admisión basado en medidas o MBAC (Measurement-Based Admisión Control) confía en las mediciones de la actual carga y en el comportamiento del tráfico (su pasado histórico) para decidir si un flujo es admitido o no, en lugar de asumir un modelo estadístico para el tráfico.

Este estudio se va a centrar sobre los algoritmos de control de la admisión basados en medidas y más concretamente en el algoritmo Measured Sum, por ser el más sencillo de implementar.

7.1.1. Algoritmo Measured Sum

Se define un período de muestreo S y una ventana T de medición, donde T corresponde a un múltiplo de S. Al final de cada período de muestreo S se miden los tráficos en los enlaces de salida y de entrada si lo hubiera. En el enlace de salida el máximo valor medido durante el tiempo T es considerado como la carga de salida. En general, si consideramos *v* como la carga medida en todas las conexiones existentes:

$$v = \sum_{j=1}^{nconexiones} L_j$$

donde L es la carga de una conexión asociada al router medido cada período de muestreo S.

Cuando una nueva aplicación genera un tráfico solicitando una nueva conexión, se inicia la medición del nuevo tráfico de entrada durante la siguiente ventana T. Al finalizar esta ventana T el valor de pico será una medida para decidir si esta nueva aplicación es admitida o no. Además el valor máximo *v* obtenido en la ventana inmediatamente anterior es considerada como la carga en el enlace de salida. Ese algoritmo se basa en:

$$v + \gamma \leq \mu B$$

donde B es el ancho de banda del enlace de salida y μ es el factor de utilización del ancho de banda B (menor que 1).

7.1.2. Algoritmo Hoeffding Bounds

Dado un flujo que denominaremos α, cuando éste solicita la admisión en la red, el algoritmo calcula su valor pico p^{α}.

El flujo será admitido si la siguiente condición se evalúa a verdadero:

$$\hat{C}_H + p^{\alpha} \leq \mu$$

donde:

$$\hat{C}_H(\hat{v}, \{p_i\}, \varepsilon) = \hat{v} + \sqrt{\tfrac{1}{2}(\ln(1/\varepsilon)\sum_i (p_i)^2}$$

es el ancho de banda equivalente de n flujos con valores pico p_i

$\hat{C}_H$ se actualiza mediante la fórmula:

$$\hat{v}' = \hat{v} + p^{\alpha}$$

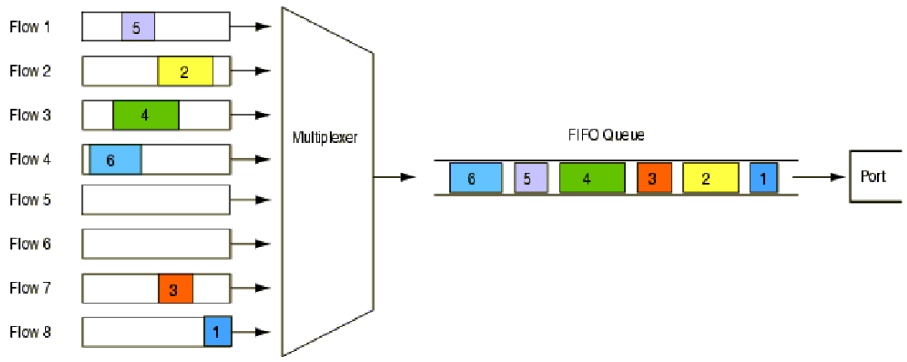
ε es un valor de pérdida especificado por el usuario, y representa la probabilidad que el valor de llegada total exceda el ancho de banda del enlace.

8. MECANISMOS DE PLANIFICACION DE LAS COLAS INTERNAS

La planificación ó Queue Scheduling determina las características temporales de los paquetes. Indica en qué momento los paquetes almacenados en las colas internas de un nodo deben ser enviados al siguiente nodo. De ahí que un planificador influye directamente en el envío de los paquetes extremo a extremo y un equivocado criterio de planificación puede originar un incremento del retardo extremo a extremo. En general, los algoritmos de planificación presentan un compromiso entre la complejidad de su implementación y el cumplimiento de las características temporales de QoS. Entre los principales algoritmos de planificación se pueden citar: *First In First Out* (*FIFO*) o *First-Come First-Served (FCFS), Priority Queue (PQ), Fair Queueing (FQ), Weighted Fair Queueing (WFQ), Weighted Round Robin (WRR) o Class Based Queue (CBQ).* Los detalles de estos algoritmos los describiremos a continuación.

8.1. First In First Out

First In First Out (FIFO) ó también llamado First-Come First Served (FCFS) es la disciplina de planificación de colas más básica que existe. En la planificación FIFO, todos los paquetes son tratados igualmente colocándolos en una cola simple y de esta manera sirviéndolos en el mismo orden en el que fueron colocados en la cola. En este tipo de planificación no hay ningún tipo de prioridad, tan sólo el orden de llegada de los paquetes.



Este mecanismo ofrece las siguientes ventajas en cuanto a su tipo de planificación:

- Para conmutadores basados en software, FIFO coloca en la cola una carga computacional extremadamente baja comparada con otros mecanismos mas elaborados.
- El mecanismo de una cola FIFO es muy predecible, los paquetes no son reordenados y el retardo máximo es determinado por la profundidad o longitud de la cola.
- FIFO proporciona una simple resolución de la contención.

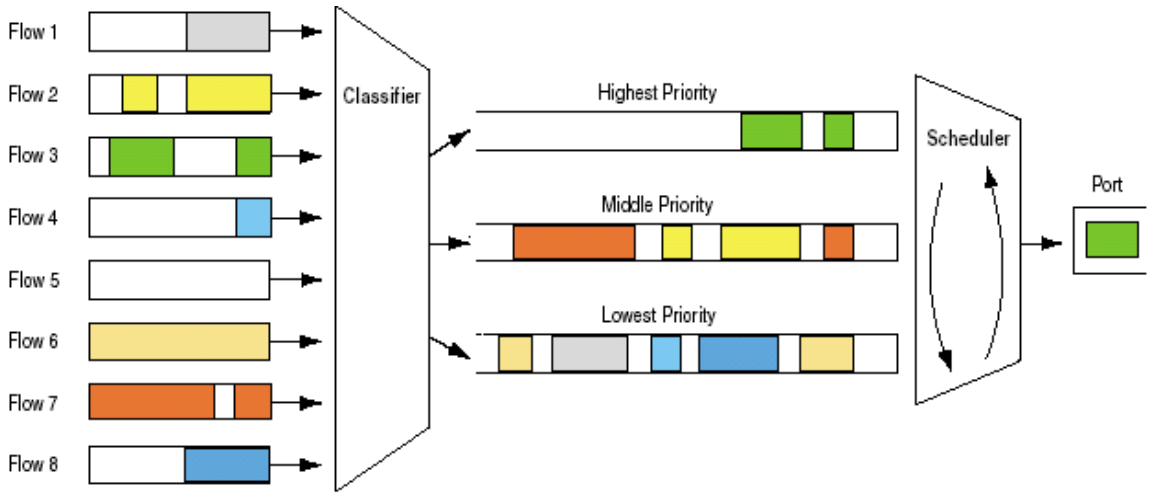
También presenta las siguientes limitaciones:

- Una simple cola FIFO no permite a los conmutadores organizar los paquetes diferenciados, es decir, no diferencia las clases de tráfico.
- Trata todos los flujos con la misma igualdad, por tanto, el retraso, pérdida de paquetes, jitter... pueden verse incrementados.
- Un determinado flujo puede consumir todo el espacio de la cola, esto puede causar que se deniegue el servicio al resto de los flujos.

8.2. Priority Queueing

Este mecanismo de planificación es una evolución o mejora del mecanismo *First In First Out*.

Los paquetes con una mayor prioridad son tratados y por lo tanto serán transmitidos antes que los paquetes de menor prioridad.



Este mecanismo requiere de una configuración de las colas. Esta configuración será la que permitirá elegir el paquete de la cola a transmitir según su prioridad.

Los beneficios de este algoritmo son los siguientes:

- Para conmutadores basados en software, PQ coloca una relativamente baja carga de computación.
- PQ permite a los conmutadores organizar los paquetes encolados. Se puede asignar una cierta prioridad a paquetes pertenecientes a un tráfico multimedia, en tiempo real, etc.

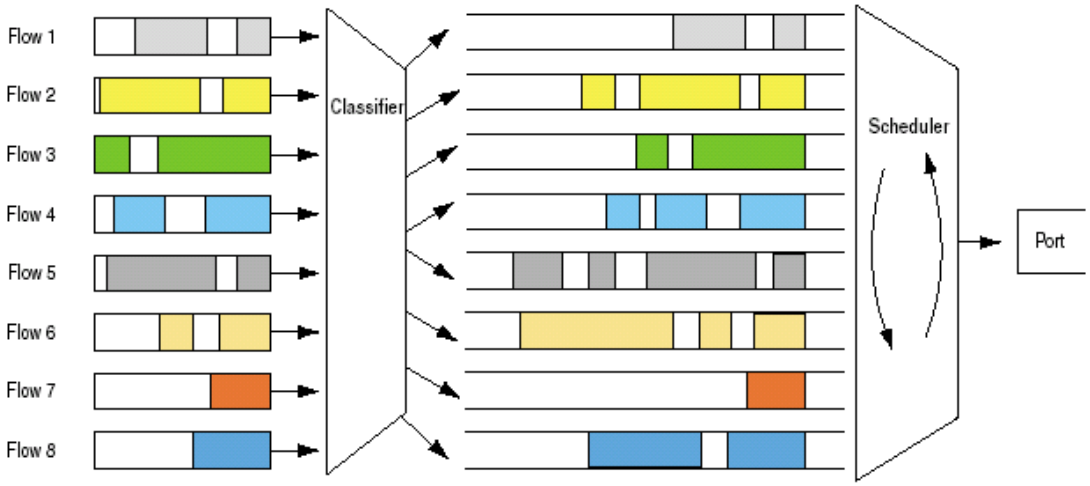
Las desventajas son las siguientes:

- Si la cantidad de tráfico con prioridad alta no está condicionada a los límites de la red, el tráfico de baja prioridad puede experimentar un excesivo retardo.
- Si el volumen del tráfico de alta prioridad es excesivo, puede provocar caídas o pérdidas de paquetes del tráfico de baja prioridad.
- Un mal comportamiento de un flujo de alta prioridad puede aumentar el retardo o jitter a otros flujos de alta prioridad debido al compartimiento de la cola.

8.3. Fair Queueing

Proporciona un asilamiento de cada flujo, da a cada flujo su propia cola y aplica el algoritmo *Round Robin* a cada una de ellas.

También proporciona imparcialidad local entre los flujos usando algoritmos de control de la congestión extremo a extremo y el mismo tamaño de paquete.



Este mecanismo presenta las siguientes desventajas:

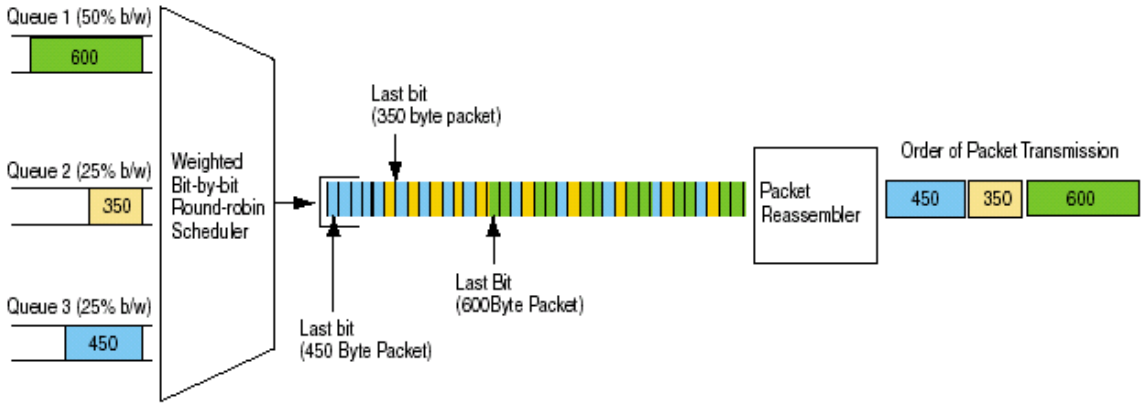
- Las implementaciones realizadas en *FQ*, son realizadas en software, no en hardware, lo que limita la velocidad de las interfaces en los límites de la red.
- El objetivo de *FQ*, es repartir el ancho de banda de manera igual para cada flujo, esto impide asignaciones variables del ancho de banda para cada flujo.
- Es sensible al orden de llegada de los paquetes.
- No proporciona un mecanismo que permita la gestión de servicios en tiempo real.

8.4. Weighted Fair Queueing

Éste mecanismo está basado en el anterior, *Fair Queueing,* y por tanto presenta algunas características comunes y a la vez ciertas mejoras.

Su principal característica es que divide el ancho de banda, al igual que su antecesor, pero este lo hace siguiendo una lógica de pesos. Con esto asegura el tiempo de respuesta en aplicaciones críticas. Además, asegura igualdad en los flujos de tráfico basados en pesos.

WFQ, da prioridad al tráfico de bajo volumen como el procedente de aplicaciones tipo *Telnet* sobre el tráfico de alto volumen como el procedente de aplicaciones tipo *FTP*.



Los beneficios que presenta el mecanismo *WFQ* son los siguientes:

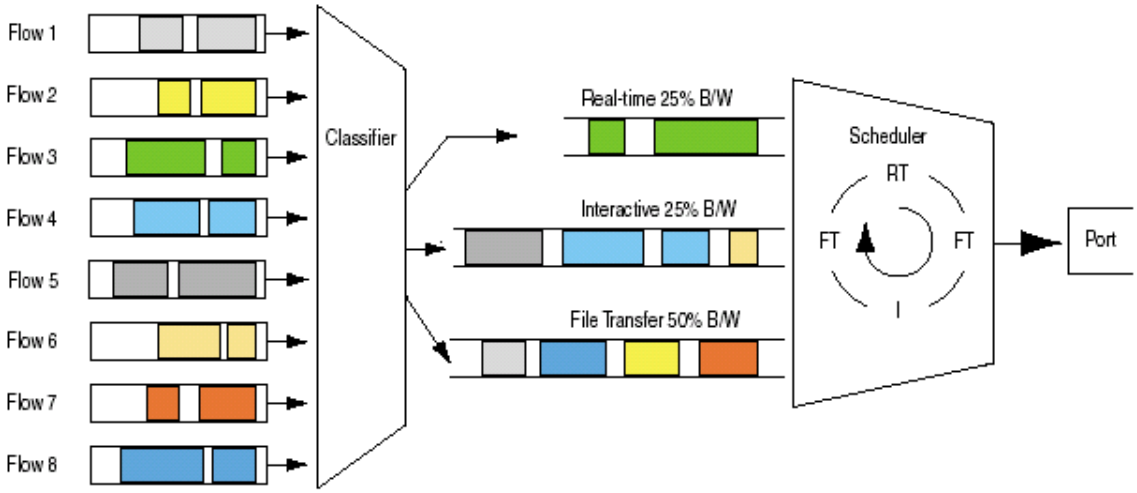
- Proporciona protección a cada clase de servicio asegurando un nivel mínimo de ancho de banda en el puerto de salida independientemente del comportamiento de otras clases de servicio.
- Cuando combinado con tráfico condicionante en los límites de la red, garantiza un ancho de banda justamente repartido a cada clase de servicio con un retraso limitado.

Las limitaciones de este mecanismo se describen a continuación:

- Como el anterior mecanismo, presenta una baja velocidad en las interfaces de los límites de la red debido a que la implementación se realiza en software y no en hardware.
- Un mal comportamiento de uno de los flujos de una clase de servicio, puede impactar en el rendimiento de otro flujo dentro de la misma clase de servicio.
- Su complejidad computacional limita la escalabilidad del sistema.

8.5. Weigthed Round Robin

En el mecanismo *Weighted Round Robin* (*WRR*) o también llamado *Class Based Queue* (*CBQ*), los paquetes son clasificados en diferentes clases de servicio y servidos a una determinada cola. Cada una de estas colas será servida en un orden determinado por un mecanismo de Round Robin.



Las ventajas de este método se describen a continuación:

- *WRR* puede ser implementado en hardware, de esta manera se puede aplicar grandes velocidades a las interfaces tanto del núcleo como de los extremos de la red.
- Asegura que todas las clases de servicio tengan acceso a al menos una cierta cantidad de ancho de banda, evitando la inanición de ancho de banda.
- Proporciona un eficiente mecanismo para ayudar al reparto de clases de servicio diferenciadas a un número razonable de flujos de tráfico altamente asociados.

La principal limitación de este método es que proporciona el correcto porcentaje de ancho de banda a cada clase de servicio solo si todos los paquetes en todas las colas tienen el mismo tamaño o cuando el tamaño del paquete es conocido por adelantado.

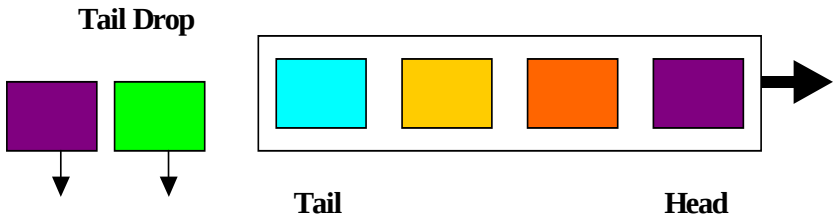
9. MECANISMOS DE GESTION DE LAS COLAS INTERNAS

Si bien es cierto que una adecuada planificación determina el cumplimiento de los requisitos de QoS solicitados por las aplicaciones, esta será eficiente si se dispone de correctos mecanismos de gestión de colas. Estos mecanismos deben anticiparse a los problemas de congestión en cada una de las colas que contienen paquetes ya clasificados, y una mala elección afectara directamente a las perdidas de paquetes. Para afrontar este reto, diferentes esquemas de *gestión de colas o buffers* han sido propuestos, pero todas ellas deben cumplir dos criterios básicos:¿ Cuando se toma la decisión para descartar los paquetes almacenados en las colas? ¿Y qué información es utilizada para tomar la decisión de descartar los paquetes? La decisión de descarte puede ser tomada a la llegada de un nuevo paquete o al comienzo de la congestión, teniendo en cuenta la granularidad de la información (por clase de servicio, por flujo, etc.). Algunos de los gestores de colas son métodos preventivos y descartan los paquetes antes de que ocurra una congestión, como es el caso de RED (Random Early Detection). En cualquier caso debe existir un compromiso entre el número de paquetes eliminados y la cantidad de retardo originado en la cola durante un periodo de congestión.

Se van a describir los siguientes gestores de cola: *Tail Drop, RED (Random Early Detection)y RIO (RED In Out)*.

9.1. Tail Drop

Cuando se produce la congestión de la red debido a que el tamaño de las colas no soporta el número de paquetes, hay que eliminar cierto número de paquetes. Este método elige como paquetes para eliminar a aquellos que se encuentran al final de la cola, es decir, los últimos en llegar.



Las ventajas de este mecanismo de gestión son las siguientes:

- Tiene una implementación muy fácil de realizar y de entender.
- Puede reducirse el número de paquetes elegidos para su eliminación.

En cambio, presenta una serie de limitaciones que hacen que no sea el más recomendable:

- Posee un comportamiento de *puerta cerrada*, es decir, si llega un paquete nuevo y la cola está saturada, este paquete no tiene oportunidad a ser transmitido.
- Es problemático para el tráfico basado en *TCP*.
- Incrementa el retraso extremo a extremo.

9.2. Random Early Detection

El *Random Early Detection (RED) es un mecanismo de gestión de colas activo o preventivo.* Activo significa que el conmutador descarta uno o más paquetes de llegada antes de que la cola de salida esté completamente llena. El resultado es que la fuente al darse cuenta, reduce la velocidad de transmisión.

El objetivo de esta gestión activa de colas es que reduce la longitud (tamaño) media de las colas en los conmutadores y por ello reduce el retardo extremo a extremo. También asegura que los recursos de la red son usados de manera más eficiente reduciendo la pérdida de paquetes que ocurre cuando las colas se saturan.

Normalmente el uso de *RED* está asociado a una utilización del mecanismo de planificación de colas tipo *FIFO* y muy efectivo para tráficos de tipo *TCP*.

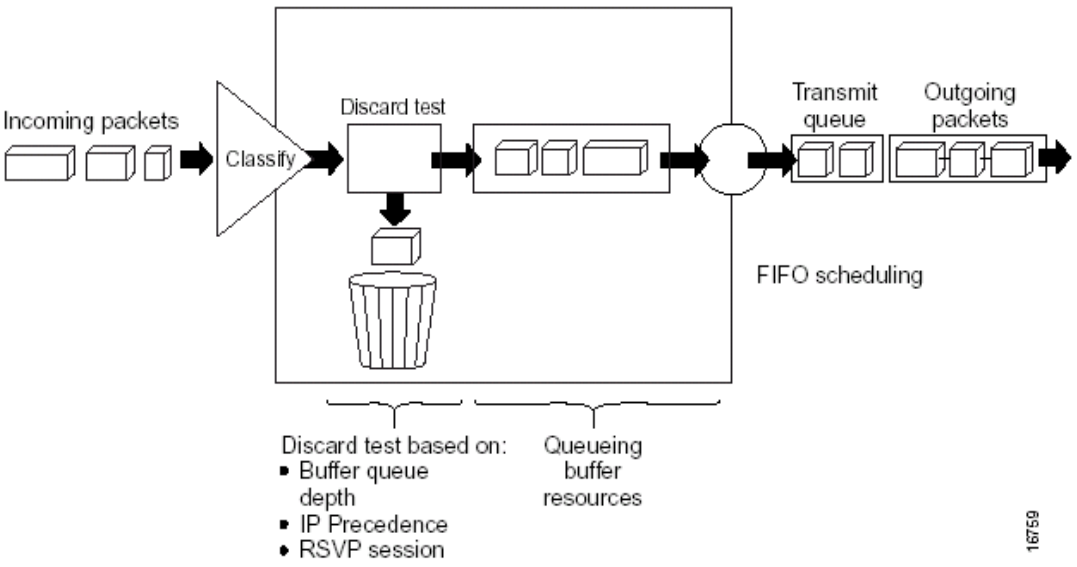
RED presenta una serie de ventajas importantes, como pueden ser:

- No provoca cambios en los actuales protocolos de la capa de transporte.
- Produce una reacción temprana al producirse la congestión, pudiéndola incluso evitar.
- El tráfico se gestiona de una manera muy eficiente.

No todo son ventajas para este método, y por tanto hay una serie de limitaciones que se describen a continuación:

- Una desventaja muy importante es su excesiva complejidad.
- No tiene efecto sobre flujos basados en un tráfico que no sea de tipo *TCP*, como podría ser *UDP*.
- La caída de paquetes empeoran los recursos de la red usados para transmitir hacia el punto de descarte.

Una modificación de este mecanismo es el *Weighted Random Detection (WRD)* en el que la diferencia es la asignación de unos pesos a los paquetes y por tanto, el descarte se hace en función de esos pesos.



9.3. RED In Out

RIO (RED In Out) usa el mismo mecanismo que *RED*, la única diferencia es que está configurado con dos conjuntos de parámetros, uno para los paquetes con perfil de entrada y otro para los paquetes con perfil de salida.

Dependiendo de estos parámetros, calcularemos los tamaños de la cola para paquetes de salida y entrada de manera diferente, es decir, mientras calculamos el tamaño de la cola de entrada, sólo los paquetes con perfil de entrada son considerados, de este modo, los paquetes con perfil de salida no incrementaran el tamaño de la cola y por lo tanto, tampoco la probabilidad de caída de esos paquetes con perfil de entrada.

El mecanismo *RIO* conlleva lo siguiente:

- Protege los paquetes de entrada de los paquetes de salida durante la congestión aunque evita las caídas innecesarias de tráfico de salida si los recursos están disponibles.
- Evita la reordenación de paquetes, como los paquetes de salida son introducidos en la cola con paquetes de entrada procedentes de la misma fuente, se mantiene la ordenación de paquetes. Los paquetes de salida pueden caerse y no ser ordenados.

10. NS NETWORK SIMULATOR

En esta sección se describirá brevemente la herramienta empleada para realizar las simulaciones llevadas a cabo en el proyecto. Se ha utilizado ns, por ser una de las herramientas para la simulación de redes más reputada dentro del mercado internacional.

Ns es un software para el funcionamiento sobre el sistema operativo *Linux* y que permite la simulación de casi cualquier tipo de red tanto redes alámbricas como redes inalámbricas. Además se pueden adaptar módulos para poder simular redes y arquitecturas de red que no soporta el software base.

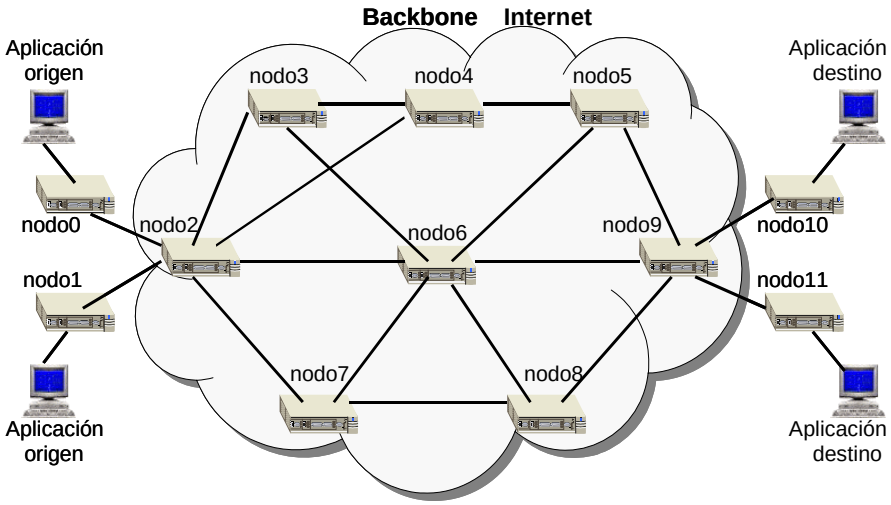
La aplicación actúa sobre unos ficheros (o scripts) de entrada que estarán codificados en el leguaje de script *TCL*, aunque en una versión mejorada que permite la utilización de clases y objetos llamado *OTCL*. Los ficheros de salida que genera la ejecución de *ns* sobre estos ficheros de entrada, serán dos: uno de ellos será un fichero de traza, que se podrá utilizar para conseguir medidas, calcular parámetros y en definitiva, generar las gráficas con el módulo de *ns, xgraph*. El otro fichero obtenido, contiene la información para que el módulo *nam*, realice una simulación gráfica del recorrido que irán haciendo los paquetes de los distintos tráficos a través de la red.

11. SIMULACIÓN DE NUEVAS ARQUITECTURAS EN INTERNET

El objetivo de este proyecto es el estudio del comportamiento de la red Internet actual basada en Ipv4 sin la utilización de elementos de calidad de servicio y las diferencias al incluir calidad de servicio en las redes. La calidad de servicio permite por ejemplo el envío de tráfico multimedia en tiempo real sin retardos considerables en la entrega.

A lo largo del presente documento exponemos diversos escenarios y para cada uno de ellos se analizan las gráficas de retardo y de pérdida de paquetes para cada uno de los gráficos.

La topología de Red que se ha empleado para la realización del estudio ha sido la siguiente:



11.1. Simulación de una gestión tipo Tail Drop

Para la simulación de este necesario hemos utilizado una topología de red con 11 nodos, dos de los cuales envían tráfico (nodos 0 y 1) y dos los reciben (nodos 10 y 11). El resto son nodos intermedios que simulan un backbone de Internet.

Los tráficos utilizados son los siguientes:

- **Tráfico CBR entre los nodos 0 y 10**. Se trata de un "flujo de bits constante" que permanece generando tráfico de forma permanente durante toda la simulación. En esta simulación hemos ido incrementando la tasa de tráfico CBR desde 1 Mbps hasta 40 Mbps. Para ello hemos aumentado la velocidad cada cierto periodo de tiempo hasta llegar a los 40 Mbps. A medida que el tráfico CBR va aumentando, aumenta también el tráfico total en los enlaces que pueden llegar al límite fijado de 50 Mbps con los consiguientes retrasos o pérdida de paquetes.

- **Tráfico Exponencial entre los nodos 0 y 10**. Este tráfico varía de forma intermitente siguiendo una distribución exponencial. Hemos fijado la tasa de paquetes en los momentos de máximo tráfico en 10 Mbps. Estos 10 Mbps con los 40 Mbps anteriores ya suman 50 Mbps, por lo que si se añade un nuevo tráfico, es muy probable que se produzcan pérdida de paquetes.

- **Tráfico FTP entre los nodos 1 y 11**. Con este tráfico simulamos una aplicación FTP. Este protocolo está construido sobre el protocolo TCP de la capa de transporte por lo que se puede observar tráfico en ambos sentidos (tráfico que se envía o recibe y los mensajes ACK de confirmación del otro extremo).

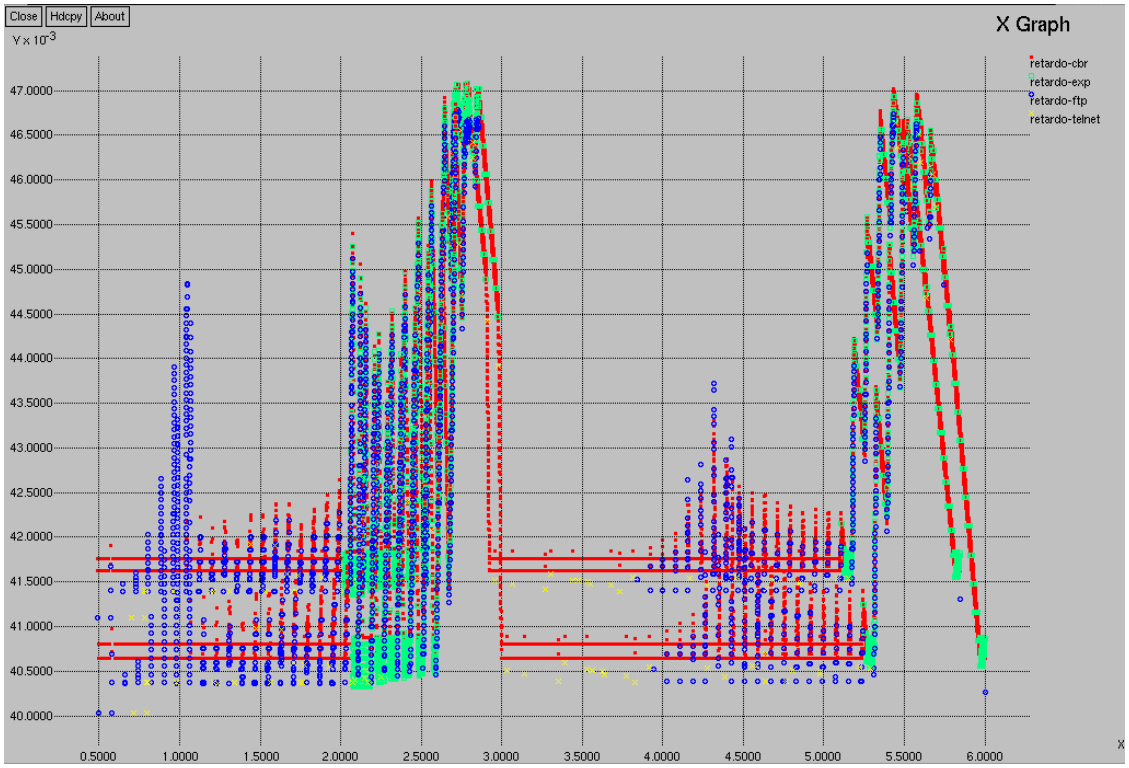
- **Tráfico Telnet entre los nodos 1 y 11**. Simulamos en este caso una aplicación Telnet. Este servicio también funciona sobre TCP por lo que podremos observar tráfico en ambos sentidos. Todos los enlaces de la topología han sido definidos como full-duplex, por que pueden transmitir tráfico en ambos sentidos de forma simultánea. El tráfico generado por Telnet es mucho menor que el de FTP debido a que Telnet únicamente transmite caracteres para acceder a terminales remotos y FTP transfiere archivos completos que pueden ocupar decenas de megabytes.

Como hemos comentado anteriormente, los enlaces utilizados en la topología simulada son de 50 Mbps y tienen un retardo de envío de paquetes entre un extremo y otro de 10 ms. El retardo de los tráficos enviados será como mínimo de 40 ms ya que cada tráfico debe atravesar como mínimo 3 routers (4 enlaces). Sin embargo se podrán producir retardos adicionales en el procesamiento de los paquetes por parte de los nodos intermedios.

En los routers intermedios hemos utilizado, para esta simulación, una gestión de colas de tipo Tail Drop. Este gestor de colas de paquetes en los routers es el más sencillo: una vez que la cola se llena, descarta los paquetes que se reciban. Es decir, no es capaz de gestionar prioridades, ni de prever cuando una cola se va a llenar, etc.

El mecanismo de planificación es de tipo FIFO, esto es, el primer paquete que llega al router es el primero que sale del router por el siguiente enlace. En ocasiones esto no es deseable, sobre todo, si quisiéramos implementar una calidad de servicio y dar prioridad a algún tipo de tráfico concreto (como pueda ser tráfico multimedia o de aplicaciones en tiempo real). Las mejoras a todas estas limitaciones se estudiarán en los siguientes escenarios de simulación.

La primera gráfica que obtenemos es para los retardos que se producen para cada tipo de tráfico. Este retardo depende de 2 componentes, uno de ellos es debido a la propagación a través del medio físico (10 ms según nuestro escenario) y el otro es depende del procesamiento de los paquetes por los nodos de la red (sobre todo del tiempo que un paquete permanece en la cola del router).

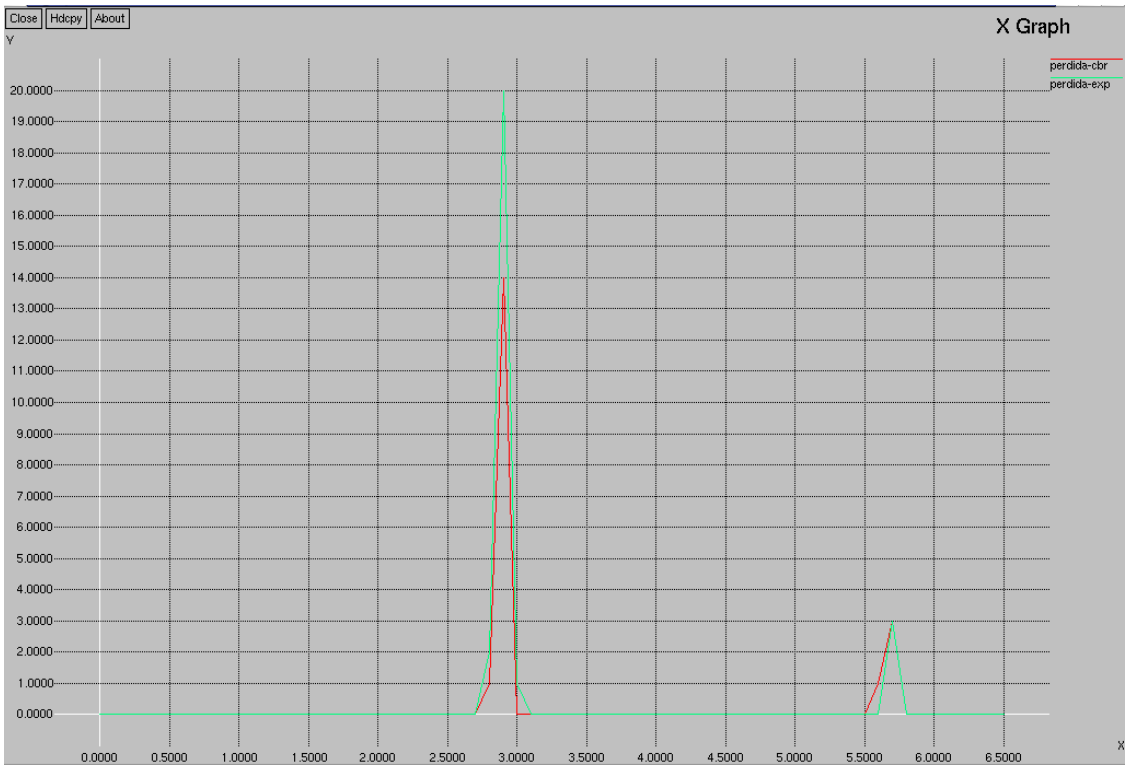


Gráfica 1. Retardo

En la gráfica podemos comprobar que se produce una saturación de tráfico aproximadamente en los instantes 2.7 s y 5.4 s de la simulación, donde se produce un mayor retardo alcanzándose picos de unos 47 ms. Esta saturación es debida a que en esos momentos existe un mayor envío de paquetes por parte de los nodos origen (coinciden todos los tipos de tráfico) y una falta de capacidad de procesamiento por parte del nodo 2 que es el nodo en el confluyen todos los tráficos.

El tráfico CBR es el tipo de tráfico que mayor retardo sufre a lo largo de la simulación. Observamos también que el resto de los tráficos sufren retardo en los instantes de mayor tráfico.

La siguiente gráfica obtenida representa los paquetes perdidos en los tipos de tráfico CBR y exponencial. Esta pérdida de paquetes viene influida por la elección del mecanismo de gestión de colas, en este caso, se ha utilizado el mecanismo Tail Drop:



Gráfica 2. Perdida de Paquetes

En la gráfica se observa que los momentos de mayor pérdida de paquetes por parte de los tráficos CBR y Exponencial, coincide con los instantes posteriores a los que el retardo alcanzaba sus picos, a partir de los instantes 2.7 s y 5.4 s. Estos instantes se corresponden con los momentos en que el tráfico exponencial está enviando paquetes, a la vez que el resto de tráficos.

Se puede comprobar que el tráfico exponencial llega a perder incluso 20 paquetes en una décima de segundo (momento de máxima pérdida de paquetes) y el tráfico CBR hasta un máximo de 15.

11.2. Simulación de una gestión tipo RED

Para esta simulación hemos reproducido los tráficos y características del apartado anterior, aunque cambiando la gestión de tipo Tail Drop por una gestión RED.

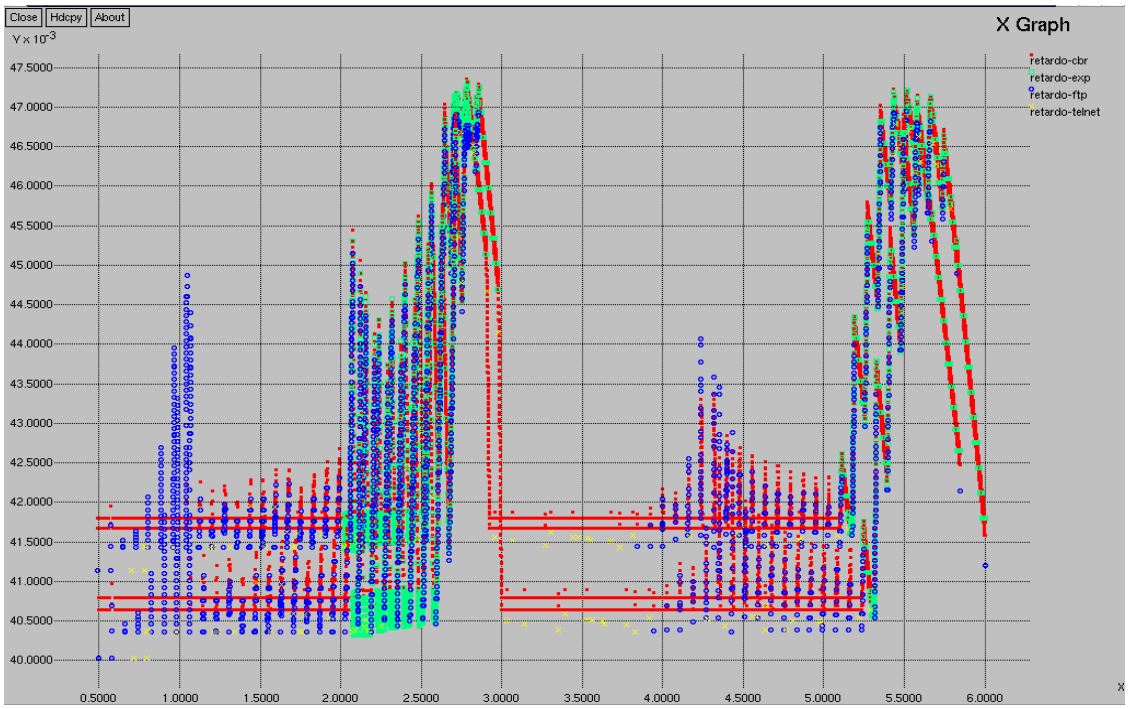
- Tráfico CBR entre los nodos 0 y 10, variándolo desde 1 Mbps hasta 40 Mbps.
- Tráfico Exponencial entre los nodos 0 y 10 de 10 Mbps máximo.
- Tráfico FTP entre los nodos 1 y 11.
- Tráfico Telnet entre los nodos 1 y 11.

Los enlaces tienen un ancho de banda de 50 Mbps y un retardo de 10 ms.

La gestión de las colas en los routers utiliza un mecanismo de tipo RED (*Random Early Detection*). Su comportamiento es distinto y más evolucionado que Tail Drop. Tiene capacidad para detectar de forma anticipada cuándo se va a llenar una cola y descarta paquetes antes de que no quede más remedio. De esta forma se evita que la cola se llene.

El planificador de las colas sigue siendo de tipo FIFO.

La gráfica de retardo obtenida para los 4 tipos de tráfico descritos con anterioridad es la siguiente:

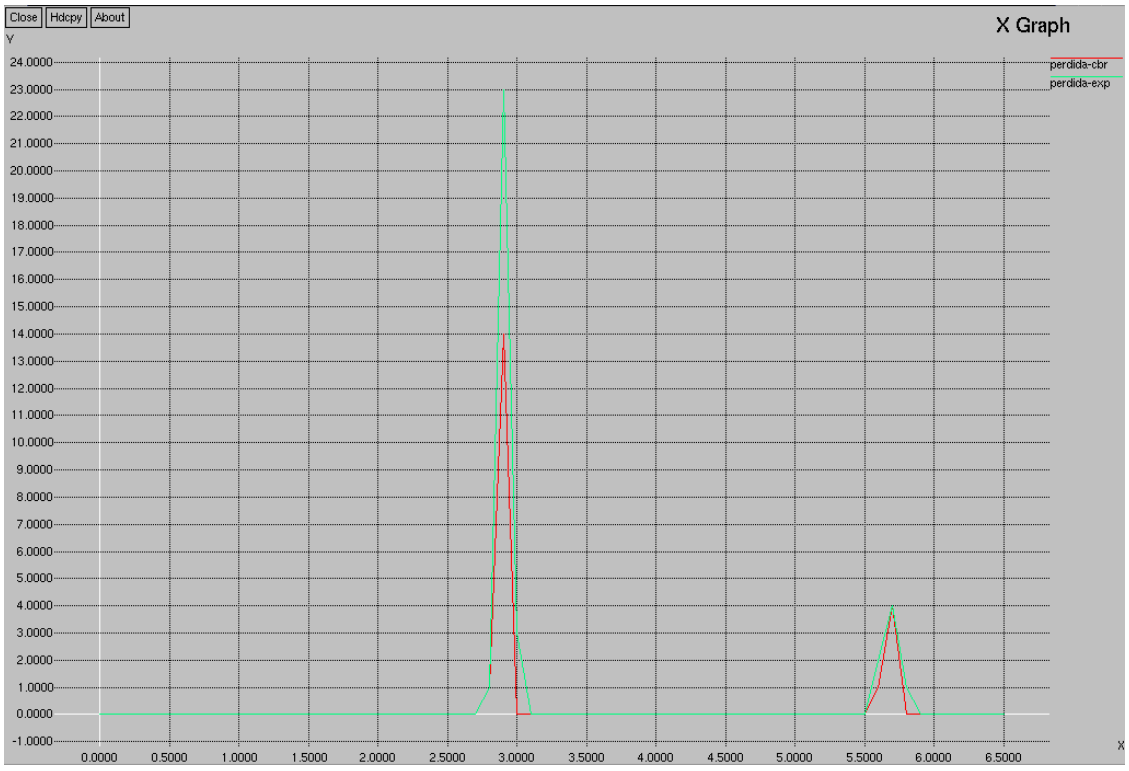


Gráfica 3. Retardo

La gráfica es muy similar a la obtenida para el escenario representado con anterioridad. Se producen máximos retardos para todos los tráficos en los mismos instantes (2.7 s y 5.4 s).

Con esto, hemos comprobado que para las condiciones planteadas en estas simulaciones no influye de manera significativa el mecanismo de gestión de colas elegido, tanto Tail Drop como RED.

La gráfica que representa la pérdida de paquetes para este escenario para el tráfico de los tipos CBR y Exponencial es la siguiente:



Gráfica 4. Perdida de Paquetes

De nuevo, se puede observar las similitudes con el anterior escenario, con lo que deducimos que para estos casos, la elección del mecanismo de gestión de colas es indiferente y se puede ratificar la gran relación guardada entre los paquetes perdidos y el retardo producido en el procesamiento de los paquetes por partes de los nodos.

11.3. Simulación de un planificador de tipo FQ

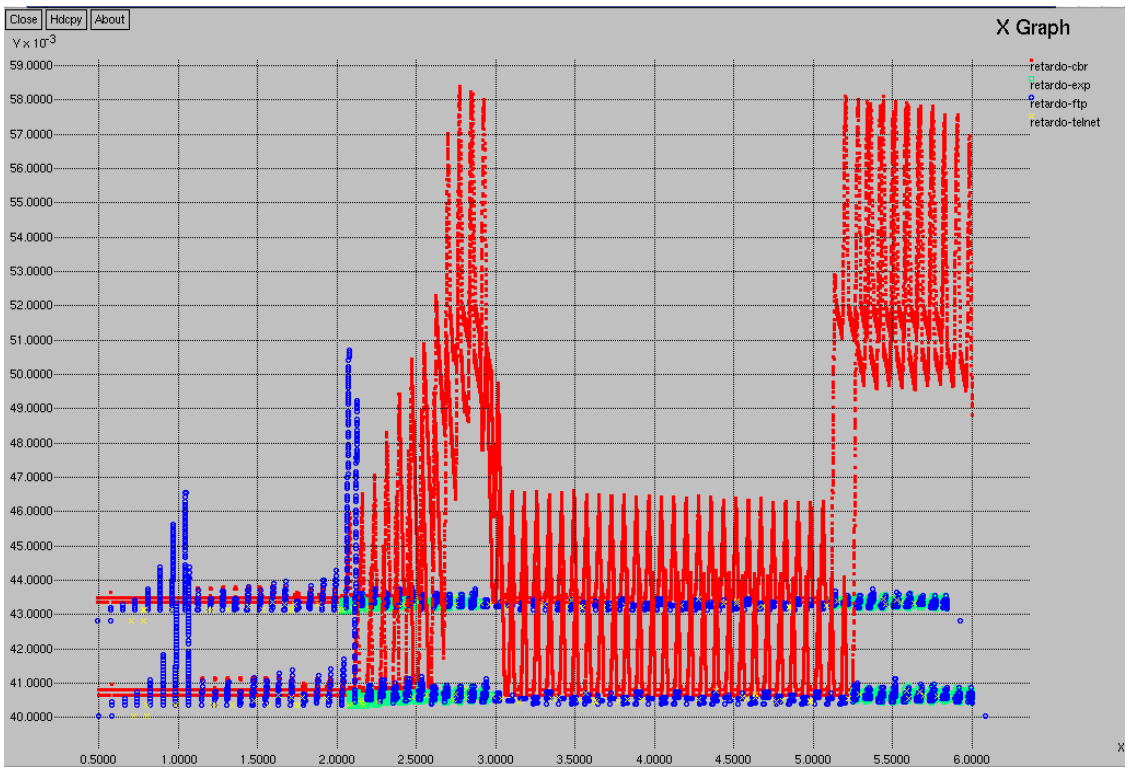
En este escenario de simulación, mantenemos la topología y las características de los tráficos de los apartados anteriores pero modificando el planificador por defecto de tipo FIFO por un planificador de cola FQ.

- Tráfico CBR entre los nodos 0 y 10, variándolo desde 1 Mbps hasta 40 Mbps.
- Tráfico Exponencial entre los nodos 0 y 10 de 10 Mbps máximo.
- Tráfico FTP entre los nodos 1 y 11.
- Tráfico Telnet entre los nodos 1 y 11.

Los enlaces tienen un ancho de banda de 50 Mbps y un retardo de 10 ms.

La planificación de las colas es de tipo FQ (Fair Queuing). Este tipo de planificador es capaz de dar mayor preferencia a unos paquetes que a otros, esto es, los paquetes que llegan más tarde podrían ser enviados antes en función del algoritmo FQ.

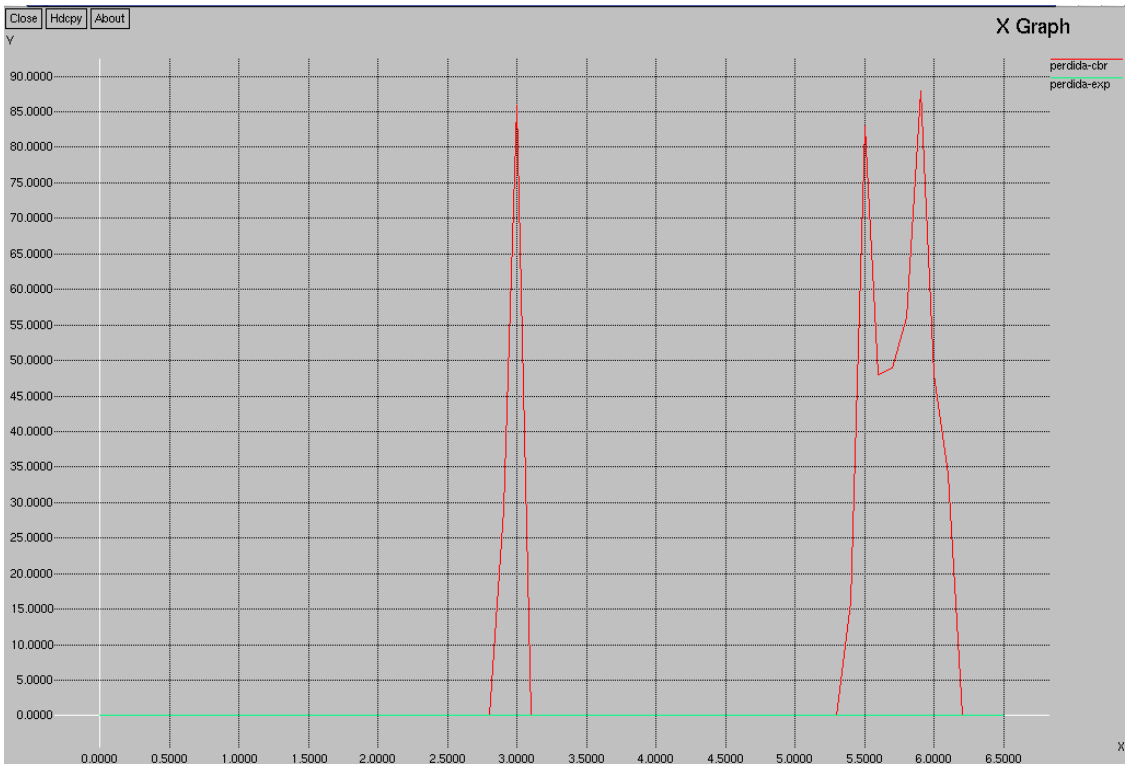
La gráfica asociada a este escenario y que describe el comportamiento de los 4 tipos de tráficos descritos con anterioridad, presenta el siguiente aspecto:



Gráfica 5. Retardo

La gráfica anterior nos indica que el único tipo de tráfico que presenta un retardo considerable es el tipo de tráfico CBR llegando a picos de 58 ms en dos ocasiones, para los instantes 2.7 s y 5.4 s que se corresponden con los instantes que los nodos origen envían más información y el nodo 2 tiene más dificultad en procesar los paquetes. Observamos que el planificador está dando preferencia a los tráficos intermitentes como el exponencial, FTP y Telnet antes que al tráfico CBR que es constante durante toda la simulación y es el que más consumo global realiza de ancho de banda.

La gráfica que representa las situaciones de pérdidas de paquetes se muestra a continuación:



Gráfica 6. Perdida de Paquetes

En este caso, se puede ver que la cantidad de paquetes perdidos también aumenta con respecto a los dos escenarios anteriores de una manera considerable pero tan sólo afecta al tráfico CBR que es el único que presenta tal pérdida de paquetes (y retardo, como se pudo comprobar en la gráfica anterior).

Se presentan tres picos que se corresponden con los momentos de mayor tráfico en la red (instantes posteriores al máximo retardo según la gráfica anterior). Se alcanzan valores de más de 85 paquetes perdidos cada 0,1 segundos.

11.4. Simulación de un planificador de tipo WRR

De nuevo continuamos con la topología de red de los apartados anteriores y las mismas características de tráfico. Queremos observar en este apartado cómo afecta el cambio del planificador de cola por un WRR (CBQ) respecto a los planificadores anteriores FIFO y FQ.

- Tráfico CBR entre los nodos 0 y 10, variándolo desde 1 Mbps hasta 40 Mbps.
- Tráfico Exponencial entre los nodos 0 y 10 de 10 Mbps máximo.
- Tráfico FTP entre los nodos 1 y 11.
- Tráfico Telnet entre los nodos 1 y 11.

Los enlaces tienen un ancho de banda de 50 Mbps y un retardo de 10 ms.

La planificación de las colas empleada se conoce como WRR (Weighter Round Robin) o también conocido como CBQ (Class Based Queue). Lo interesante de este planificador de colas es que podemos definir distintas clases de tráfico y para cada clase asignarle una cola distinta con características concretas.

En el único nodo en que hemos utilizado este planificador es en el 2, que es que tiene un mayor tráfico. Además sólo se ha considerado el tráfico saliente puesto que es el mayor. En el resto de los nodos es indiferente utilizar un planificador u otro puesto que el tráfico no se acumula en las colas de los routers.

El planificador CBQ nos permite por ejemplo reservar ancho de banda para tráficos concretos. Por ejemplo, se podría reservar 10 Mbps para el tráfico exponencial de tal forma que cuando se envíe este tipo de tráfico tenga ancho de banda suficiente y no se genere un retardo adicional. Durante el tiempo en que no se envía este tráfico exponencial se puede configurar si queremos que su ancho de banda lo utilice otro tráfico o por el contrario, que no lo utilice nadie.

También es posible asignar prioridades a los tráficos. Aquellos tráficos con menor prioridad esperarán en la cola hasta que los tráficos con mayor prioridad sean enviados. El tratamiento de prioridades es interesante para ofrecer una adecuada calidad de servicio en función de los tráficos procesados. Aquellos tráficos para aplicaciones multimedia o de tiempo real deberán ser configurados con una mayor prioridad y se les debe reservar un porcentaje sobre al ancho de banda máximo del enlace.

El planificador CBQ es altamente configurable y nos permite optimizar el tráfico que deseemos en nuestro escenario.

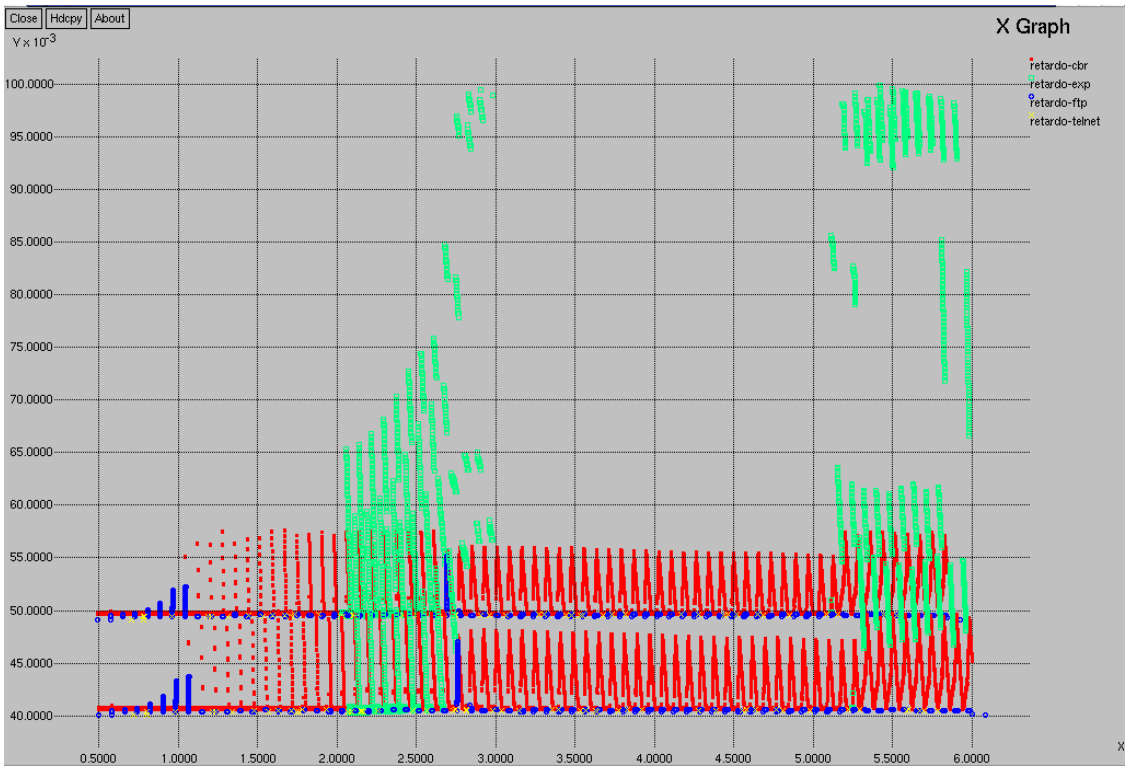
El escenario representado viene determinado por el fragmento de código:

```
#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
$class1 setparams $topclass true 0.7 auto 7 1 0

#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass true 0.3 auto 8 1 0

#FTP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.1 auto 6 1 0

#Telnet
set class4 [new CBQClass]
set queue4 [new Queue/RED]
$class4 install-queue $queue4
$class4 setparams $topclass true 0.1 auto 5 1 0
```

Asignamos distintas prioridades de tráfico. De mayor a menor: Telnet, FTP, CBR y Exponencial. El tráfico exponencial es el de menor prioridad por lo que tendrá los mayores retardos y pérdidas de paquetes.

Repartimos el caudal de 50 Mbps del enlace en: 70% máximo para CBR, 30% máximo para Exponencial, 10% máximo para FTP, y 10% máximo para Telnet.

La gráfica de retardos en este escenario se muestra a continuación:

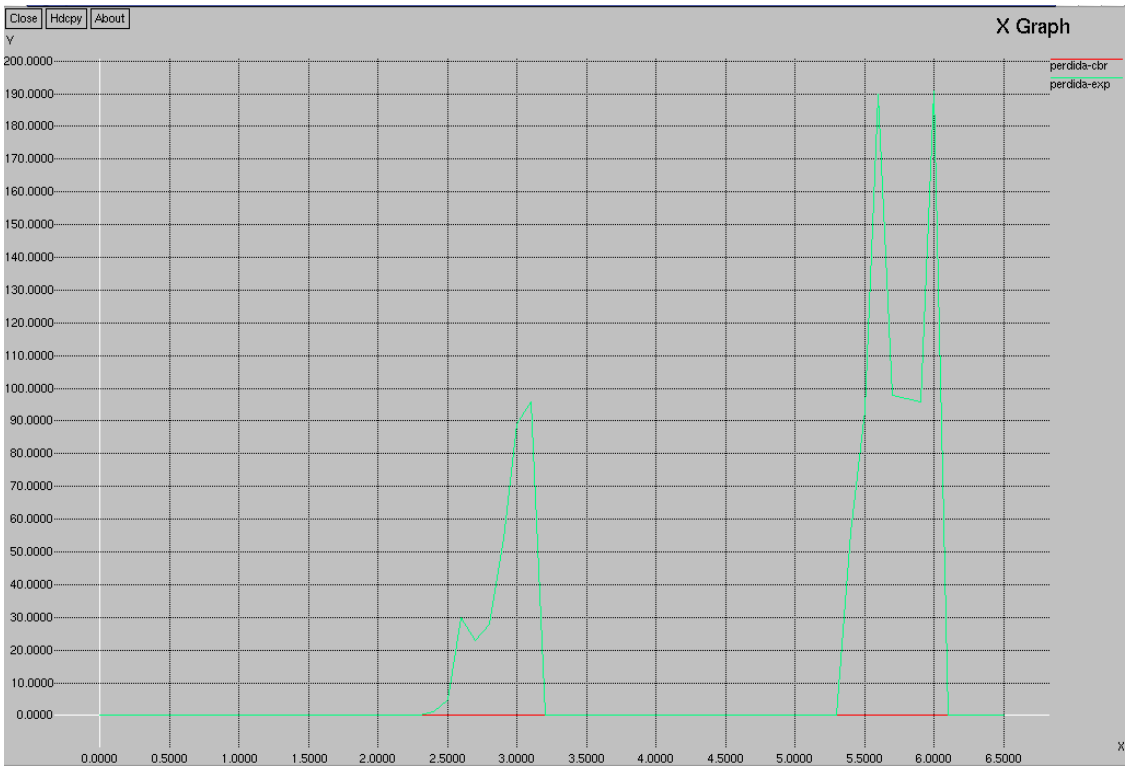


Gráfica 7. Retardo

Se puede verificar como el tipo tráfico que más retardo sufre es el tipo Exponencial, alcanzando valores muy altos de incluso 100 ms. El tráfico CBR tiene un menor retardo, el cual aumenta desde el principio de la gráfica conforme se incrementa el tráfico CBR. Sin embargo, no llega a tardar más de 60 ms.

Como vemos, en este escenario, hemos optimizado los retardos de los tráficos FTP y Telnet y, en menor medida, del tráfico CBR.

La gráfica de pérdida de paquetes se muestra a continuación:



Gráfica 8. Perdida de Paquetes

Se observa de nuevo la relación entre la pérdida de paquetes y el retardo, la gráfica detalla una gran pérdida de paquetes para el tipo de tráfico Exponencial pero por contrario ninguna pérdida para el tipo de tráfico CBR. Comprobamos que el tráfico CBR puede llegar con retraso al destino aunque no se pierden paquetes.

Esta pérdida de paquetes para el tipo de tráfico Exponencial llega a alcanzar valores de hasta casi 200 paquetes, que es ya una cifra importante en cuanto a la calidad de servicio se refiere.

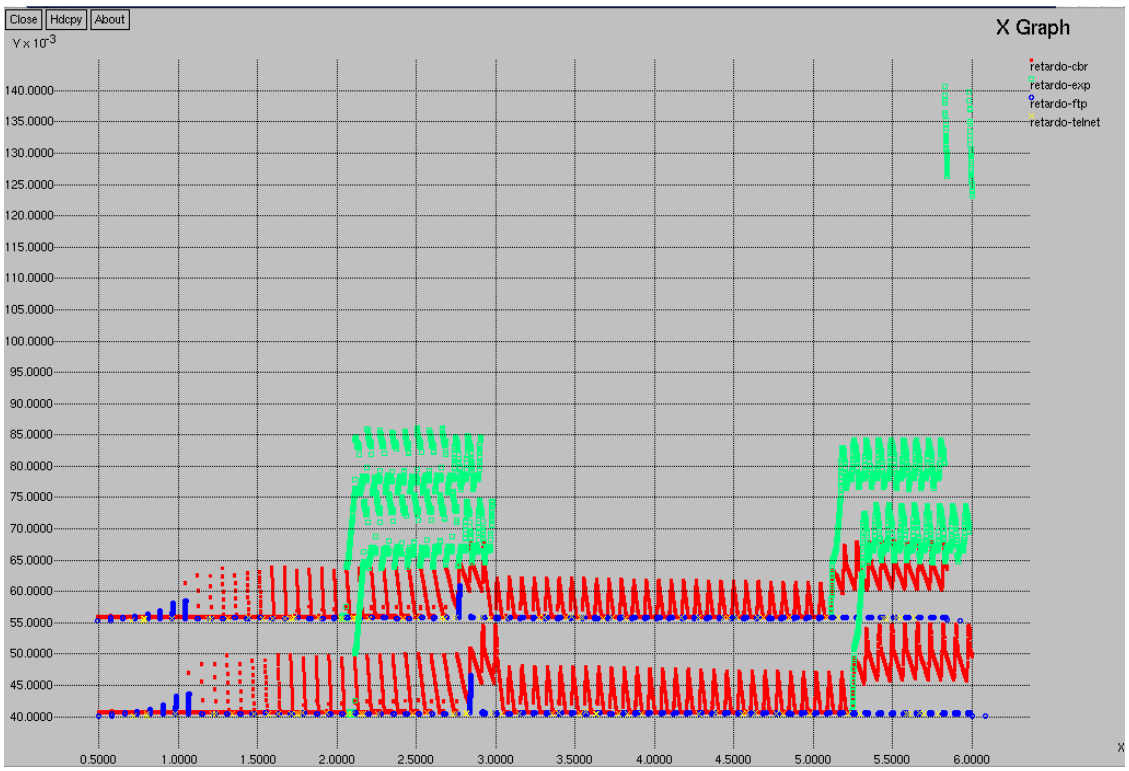
Vamos a preparar ahora un segundo escenario en el cual demos una menor prioridad a los tráficos Expo y CBR por un lado y una mayor prioridad a los tráficos FTP y Telnet. Queremos optimizar la calidad de servicio para estos dos últimos tráficos, a costa de los tráficos Expo y CBR. Al contrario que en el caso anterior, deseamos que no sólo se vea afectado el tráfico Exponencial sino también el CBR.

Los parámetros de configuración son:

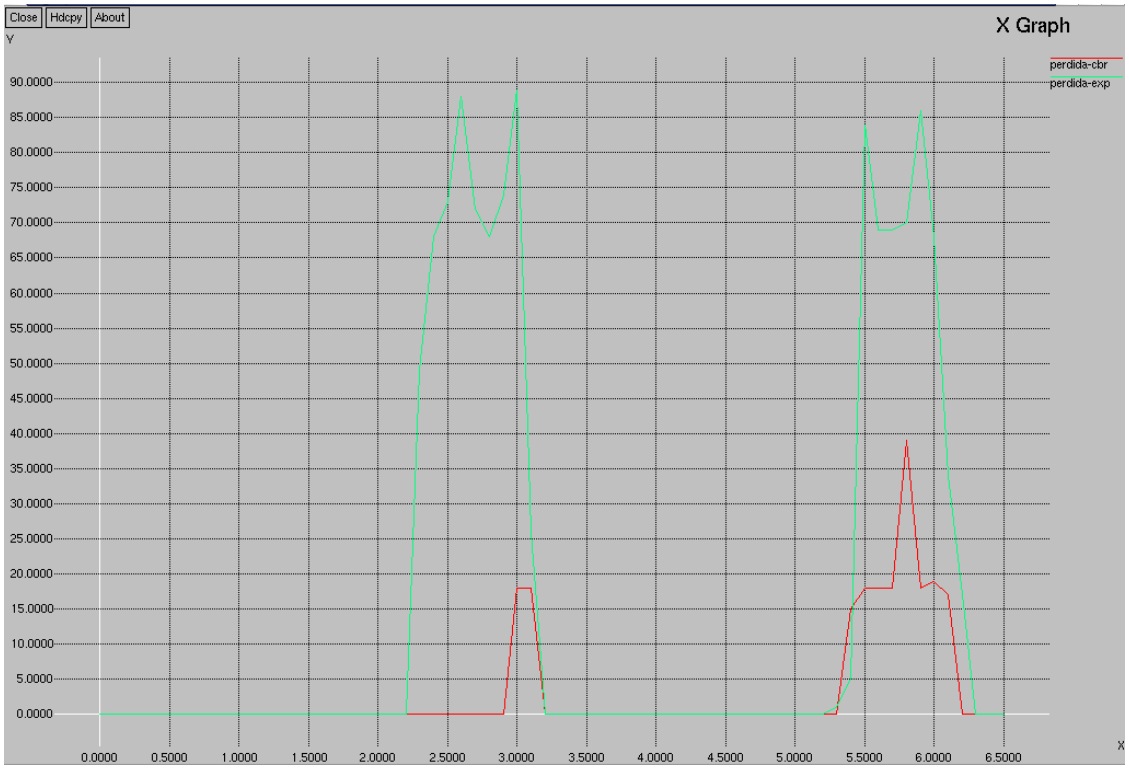
```
#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
$class1 setparams $topclass true .7 auto 7 1 0

#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass false .15 auto 7 1 0

#FTP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.1 auto 5 1 0

#Telnet
set class4 [new CBQClass]
set queue4 [new Queue/RED]
$class4 install-queue $queue4
$class4 setparams $topclass true 0.1 auto 5 1 0
```

Para el tráfico CBR permitimos que use hasta un 70% del canal (necesitaría un 80% para enviar 40 Mbps) y para el tráfico Exponencial, un 15% máximo (necesitaría un 20% para enviar sus 10 Mbps). Permitimos que el ancho de banda ocupado por CBR pueda crecer si las circunstancias de la red lo permiten, pero bloqueamos el Exponencial a un 15% aunque el resto del canal no se utilice. Esto ocasiona que siempre que el tráfico Exponencial está generando paquetes, una parte de ellos se perderá (2,5 Mbps). El tráfico CBR perderá paquetes siempre que se esté enviando simultáneamente con el resto de tráficos.

Las gráficas de este segundo escenario de retardo y pérdida de paquetes se muestran a continuación:



Gráfica 9. Retardo



Gráfica 10. Perdida de Paquetes

11.5. Conclusiones

Según las gráficas y los escenarios de simulación de los apartados anteriores, concluimos que para incorporar una adecuada calidad de servicio a los tráficos es necesario un planificador como CBQ en el cual podamos definir comportamientos diferentes en función del tipo de tráfico. Cada uno de los tráficos se puede configurar con un nivel de prioridad distinto y para cada uno de ellos se puede reservar cierto ancho de banda en los enlaces.

El planificador FIFO no es apropiado puesto que los paquetes menos prioritarios estarían entorpeciendo a aquellos más prioritarios (no distingue entre distintos tráficos). El planificador FQ mejora el comportamiento de FIFO aportando una planificación automática en función de los tipos de tráficos. Según hemos observado en las gráficas, FQ da prioridad a los tráficos intermitentes o que ocupan un menor ancho de banda global frente a aquellos que hacen un uso continuado de los recursos de la red.

Los gestores de cola que hemos utilizado han sido Tail Drop y RED. El primero descarta los nuevos paquetes cuando la cola está llena y el segundo, descarta paquetes de forma aleatoria cuando detecta que la cola se está llenando. A pesar de esta diferencia de funcionamiento, apenas hemos notado diferencias en las gráficas de los dos tipos de gestores de cola.

Concluimos por tanto que para aportar una adecuada calidad de servicio, los routers deben ofrecer comportamientos diferentes en función del tipo de tráfico. Los paquetes más prioritarios deben estar etiquetados, mediante un código en un campo de prioridad, para que los routers puedan procesar estos paquetes antes que el resto o bien, tengan capacidad para detectar los paquetes más prioritarios (según el protocolo usado, por ejemplo). Los planificadores de tipo FIFO y los gestores como Drop Tail, utilizados habitualmente en redes bajo la versión actual IP, Ipv4, no son suficientes para implementar calidad de servicio.

11.6. Simulación del tráfico generado por una aplicación multimedia

En este escenario de simulación, mantenemos la topología de los apartados anteriores pero incluyendo un nuevo tráfico que representa una aplicación en tiempo real, lo que permitirá poner a prueba la calidad de servicio de la red.

La aplicación en tiempo real escogida es la emisión de una película por parte de un nodo y la recepción de la misma en otro.

Para una mayor claridad en las gráficas únicamente utilizaremos 3 tipos de tráficos, según explicamos a continuación:

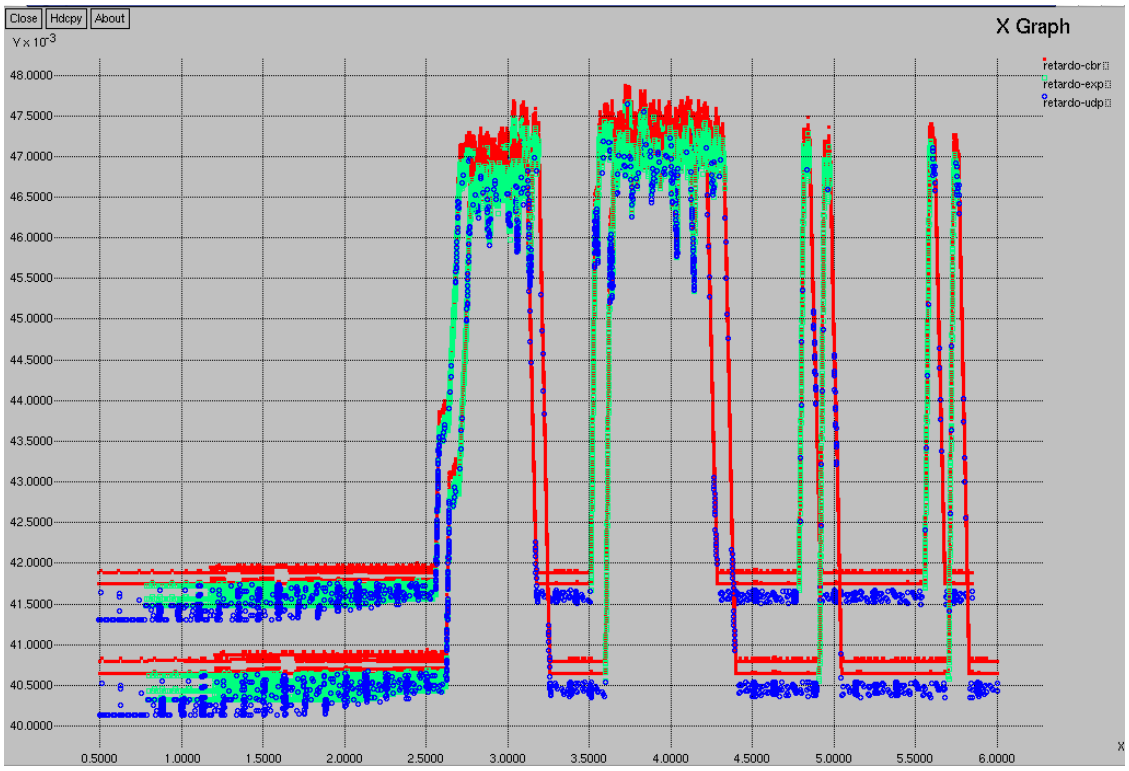
- Tráfico CBR entre los nodos 0 y 10, variándolo desde 1 Mbps hasta 45 Mbps.
- Tráfico Exponencial entre los nodos 0 y 10 de 15 Mbps máximo.
- Tráfico UDP generado por la película Star Wars emitida en tiempo real.

Los enlaces tienen un ancho de banda de 50 Mbps y un retardo de 10 ms.

Hemos utilizado la gestión de cola tipo RED (Random Early Detection) y un planificador FIFO. Como hemos comentado anteriormente, este tratamiento en los routers no incluye calidad de servicio.

Para incluir calidad de servicio sería recomendable utilizar un planificador CBQ asignando máxima prioridad al tráfico de la película, como haremos posteriormente.

Observamos el resultado sin calidad de servicio:

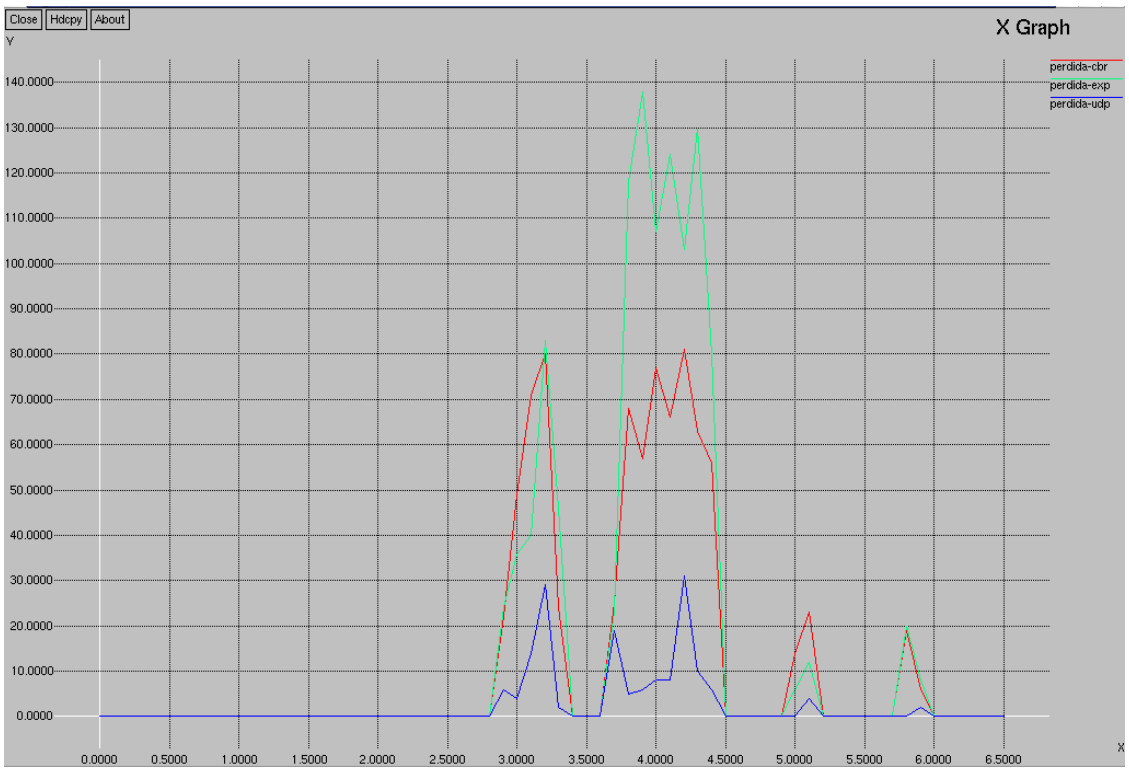


Gráfica 11. Retardo

En esta gráfica observamos claramente cómo los 3 tipos de tráfico sufren retardos. No se optimiza ninguno de ellos, sino que los tres se tratan por igual. En el momento en que se envía tráfico exponencial, se genera una sobrecarga en los enlaces que se traduce en constantes retardos de todos los tráficos (picos de 48 ms).

El tráfico UDP, representante de la película Star Wars, también sufre retardo considerable. Esto se podría traducir en términos menos abstractos como que bastantes fotogramas pertenecientes a la película se retrasarían y por tanto, la película se visualizaría con interrupciones de imagen para estos momentos.

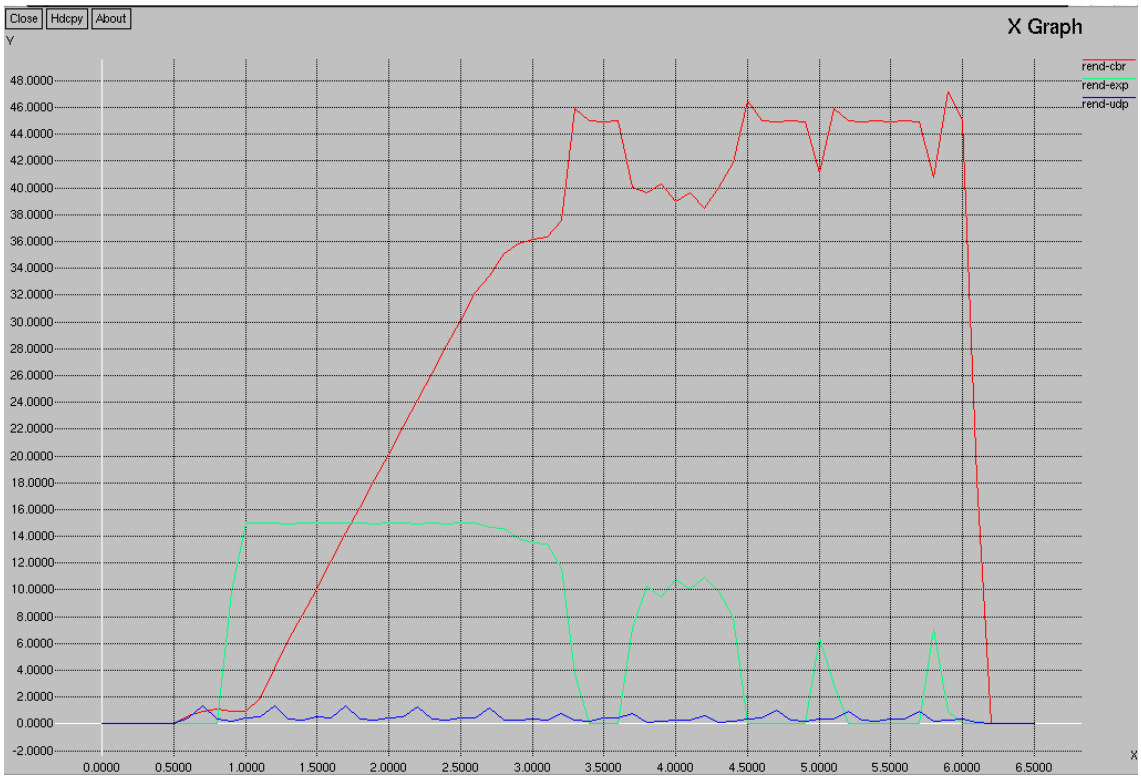
El gráfico que muestra la pérdida de paquetes para esta nueva situación en la que los tipos de tráfico a mostrar son CBR, Exponencial y UDP sigue a continuación:



Gráfica 12. Perdida de Paquetes

En esta situación, lo verdaderamente reseñables la pérdida de esos paquetes para el tipo de tráfico UDP, es decir, los correspondientes o generados por la emisión de la película. Esta pérdida de paquetes provocaría una falta de fotogramas o fragmentos de la película en la recepción y por tanto ésta no se visualizaría de manera íntegra. Observamos, que se producen pérdidas de paquetes también para el resto de los tráficos.

En esta última gráfica mostramos el rendimiento de los paquetes (Mbps recibidos de cada tráfico). Observamos cómo se incrementa el tráfico CBR desde 1 Mbps hasta 45 Mbps y después trata de permanecer constante en 45 Mbps. Sin embargo, cuando se genera tráfico exponencial, el tráfico CBR se ve disminuido en velocidad, ya que ambos tienen que repartirse, junto al tráfico UDP, el canal de 50 Mbps, que resulta muy escaso para los 45 Mbps CBR, 15 Mbps Expo y casi 2 Mbps de UDP.

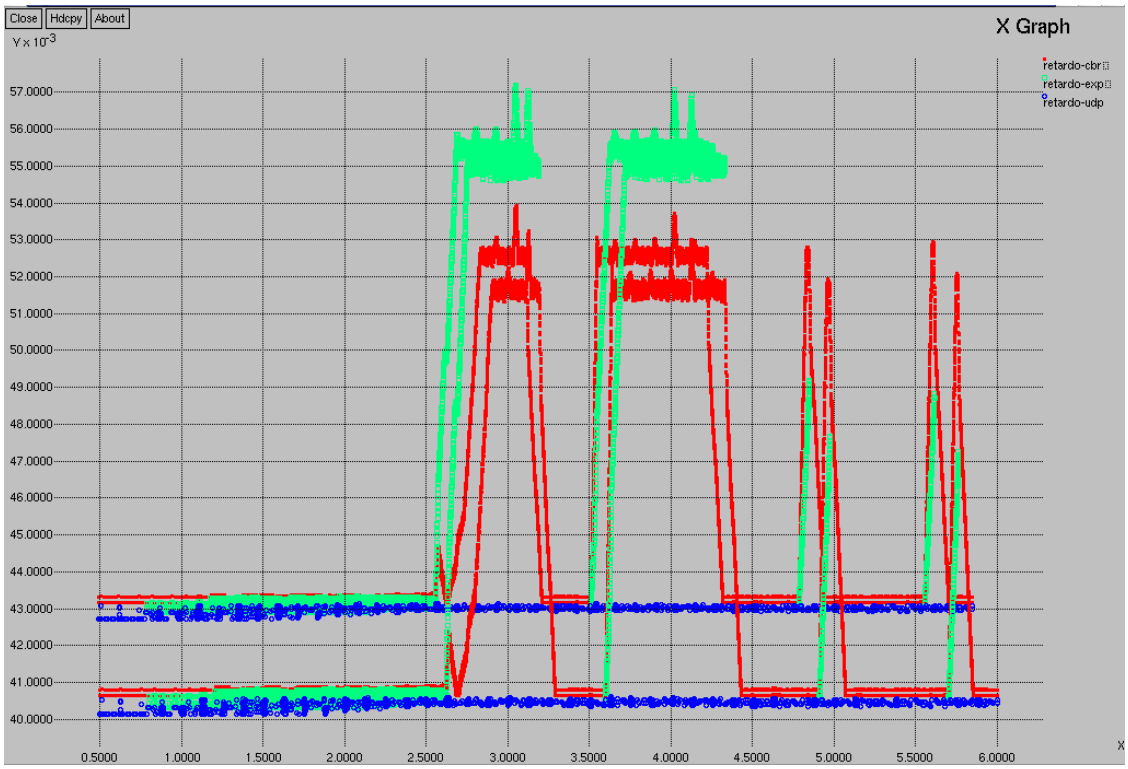


Gráfica 13. Rendimiento

A continuación vamos a modificar el escenario anterior para incluir una adecuada calidad de servicio en el tráfico multimedia. Deseamos por tanto optimizar a toda costa el tráfico UDP, correspondiente a la traza de Star Wars, de forma que no sufra retardos ni pérdida de paquetes considerables. Para este escenario, preparamos un planificador CBQ que asigne máxima prioridad al tráfico UDP:

```
#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
$class1 setparams $topclass true 0.8 auto 7 1 0

#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass true 0.3 auto 7 1 0

#UDP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.2 auto 5 1 0
```

En esta configuración del planificador asignamos mayor prioridad al tráfico UDP respecto a los tráficos CBR y Expo. Además, le reservamos un ancho de banda de 10 Mbps (20% del canal) lo cual es mucho más de lo que necesita este tráfico. Esto se traduce en prioridad absoluta para el tráfico multimedia que no esperará en las colas de los routers sino que se enviará inmediatamente, aunque para ello haya que descartar el resto de los tráficos.

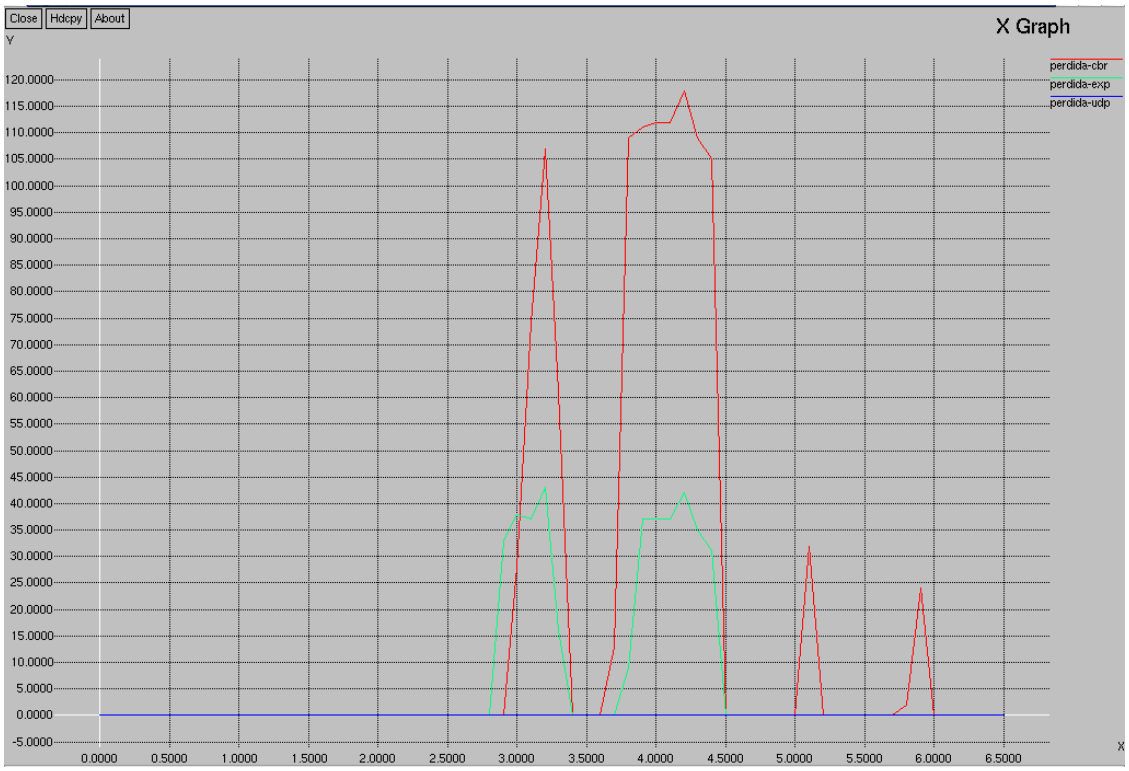


Gráfica 14. Retardo

Vemos que el retardo de UDP es mínimo y constante independientemente del resto de los tráficos. Esto garantiza que se recibirá la película de forma continua en el destino sin verse afectada por el resto de tráficos del canal.

Por el contrario, los tráficos CBR y Expo se ven muy afectados, incrementándose de forma notable su retardo, el cual llega a alcanzar valores de 57 ms (frente a los 48 ms del escenario anterior).

La gráfica que muestra la pérdida de paquetes es la siguiente:

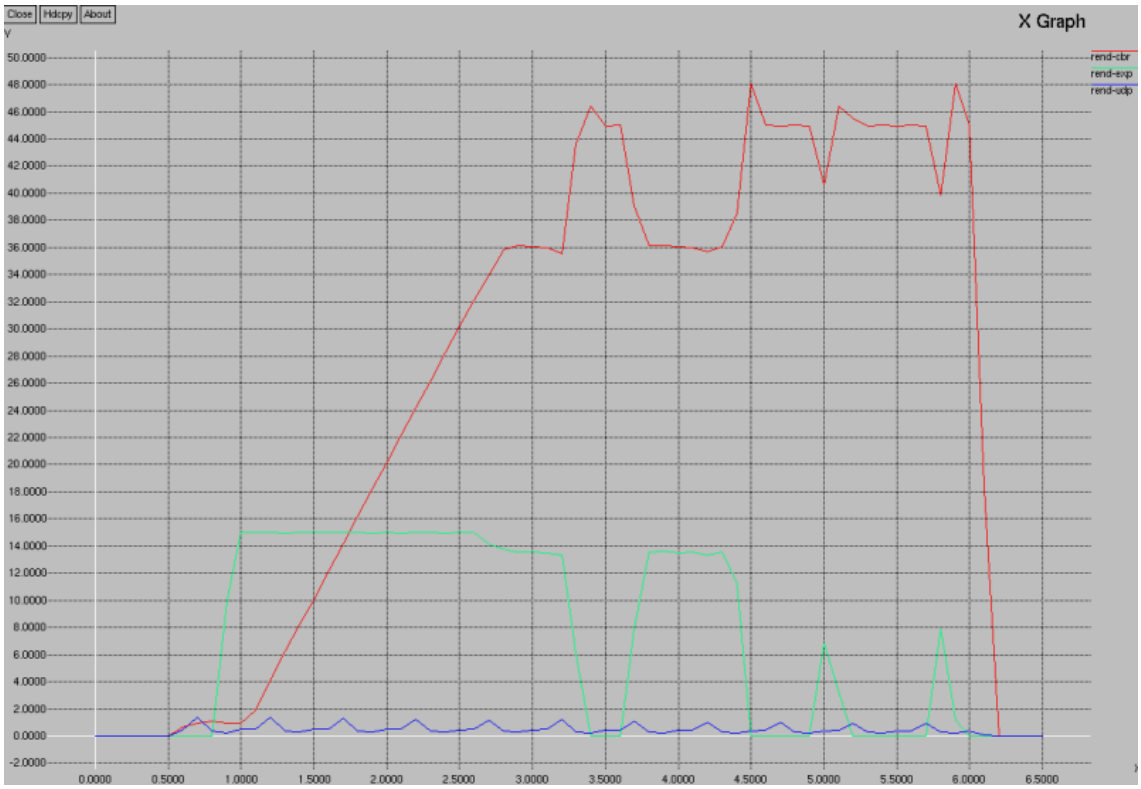


Gráfica 15. Perdida de Paquetes

Vemos que no se pierde ningún paquete del tráfico multimedia, lo cual es justamente lo que deseamos conseguir. Sin embargo, el resto de los tráficos se encuentran muy afectados cuando se envían simultáneamente todos a la vez (50 Mbps de los enlaces es insuficiente).

También podríamos haber modificado ligeramente el planificador para que permita alguna pérdida de paquetes UDP y reduzca la pérdida de paquetes del resto de los tráficos. Esto se traduciría en destino como la pérdida de algún frame, lo cual no es excesivamente problemático siempre que no se dispare el retardo.

Finalmente incluimos la gráfica del rendimiento para este segundo escenario. El tráfico UDP tiene garantizados 10 Mbps por lo que siempre podrá enviar todos sus datagramas. El resto de los tráficos se reparten el resto del canal que el tráfico multimedia no está utilizando (de forma que no se supere el límite de 50 Mbps sumando todos los tráficos en cada instante).



Gráfica 16. Rendimiento

12. LISTADO DE CÓDIGO

Script para el cálculo del retardo: retardo.awk

```
BEGIN {
  # simple awk script to generate end-to-end packet lifetime statistics
  # in a form suitable for plotting with xgraph.
  # Lloyd Wood, July 1999.
  # http://www.ee.surrey.ac.uk/Personal/L.Wood/ns/

  highest_packet_id = 0;
}

{
  action = $1;
  time = $2;
  node_1 = $3;
  node_2 = $4;
  src = $5;
  flow_id = $8;
  node_1_address = $9;
  node_2_address = $10;
  seq_no = $11;
  packet_id = $12;


  if (flow_id == FID) {
    if ( packet_id > highest_packet_id ) highest_packet_id = packet_id;

    # getting start time is not a problem, provided you're not starting
    # traffic at 0.0.
    # could test for sending node_1_address or flow_id here.
    if ( start_time[packet_id] == 0 )  start_time[packet_id] = time;

    # only useful for small unicast where packet_id doesn't wrap.
    # checking receive means avoiding recording drops
    if ( action != "d" ) {
      if ( action == "r" ) {
        # could test for receiving node_2_address or flow_id here.
        if (node_2_address == DST) end_time[packet_id] = time;
      }
    } else {
      end_time[packet_id] = -1;
    }
  }
}
END {
   for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
     start = start_time[packet_id];
```

```
        end = end_time[packet_id];
        packet_duration = end - start;

        if ( start < end ) printf("%f %f\n", start, packet_duration);
    }
}
```

Script de Linux para cálculo del retardo

```
grep "tcp" out.tr > tcp.tr
grep "exp" out.tr > exp.tr
grep "cbr" out.tr > cbr.tr
sleep 1m
awk -f retardo.awk -v FID=1 DST=10.0 cbr.tr > retardo-cbr
awk -f retardo.awk -v FID=2 DST=10.1 exp.tr > retardo-exp
awk -f retardo.awk -v FID=3 DST=11.0 tcp.tr > retardo-ftp
awk -f retardo.awk -v FID=4 DST=11.1 tcp.tr > retardo-telnet
sleep 1m
xgraph retardo-cbr retardo-ftp retardo-exp retardo-telnet -geometry 800x400
```

Simulación de una gestión tipo Tail Drop

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]

$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}

#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(8) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(10) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 50Mb 10ms DropTail

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
```

```
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down


#Define a 'finish' procedure
proc finish {} {
   global ns nf tf f1 f2
#Close the output files
   $ns flush-trace
   close $nf
   close $tf
   close $f1
   close $f2

#Call xgraph to display the results
   exec nam out.nam &
   exec xgraph perdida-cbr perdida-exp perdida-ftp perdida-telnet -geometry 800x400 &

   exit 0
}

proc varia_CBR {} {
      global trafico1 tasa1

         #Get an instance of the simulator
         set ns [Simulator instance]
         #Set the time after which the procedure should be called again

      set tasa1 [expr $tasa1+2000000]

   # Va incrementando la tasa hasta 40Mbps
   if {$tasa1<=40000000} {
         $trafico1 set rate_ $tasa1
      set time 0.1
      set now [$ns now]
      $ns at [expr $now+$time] "varia_CBR"
   }

}
```

```
proc record {} {
        global final1 final1e f1 f2
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final1e set nlost_]

           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"

           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final1e set nlost_ 0

           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}


# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1
$ns attach-agent $n(10) $final1

set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen1e [new Agent/UDP]
set final1e [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1e
$ns attach-agent $n(10) $final1e

set trafico1e [new Application/Traffic/Exponential]
$trafico1e set packetSize_ 500
$trafico1e set rate_ 10000000
$trafico1e set burst_time_ .5
$trafico1e set iddle_time_ .1
$trafico1e attach-agent $origen1e
```

```
$ns connect $origen1e $final1e

# Trafico desde el nodo 1 al nodo 11  FTP
set origen2 [new Agent/TCP/FullTcp]
set final2 [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2
$ns attach-agent $n(11) $final2

# set up TCP-level connections
$final2 listen
$origen2 set window_ 100

set trafico2 [new Application/FTP]
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Trafico desde el nodo 1 al nodo 11  TELNET
set origen2t [new Agent/TCP/FullTcp]
set final2t [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2t
$ns attach-agent $n(11) $final2t

# set up TCP-level connections
$final2t listen
$origen2t set window_ 100

set trafico2t [new Application/Telnet]
$trafico2t set interval_ .1
$trafico2t attach-agent $origen2t

$ns connect $origen2t $final2t


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen1e set class_ 2
$ns color 2 Blue
$origen2 set class_ 3
$ns color 3 Green
$origen2t set class_ 4
$ns color 4 Brown


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
```

```
$ns at 0.5 "$trafico1e start"
$ns at 0.5 "$trafico2 start"
$ns at 0.5 "$trafico2t start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico1e stop"
$ns at 6 "$trafico2 stop"
$ns at 6 "$trafico2t stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación de una gestión tipo RED

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]

$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}

#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms RED
$ns duplex-link $n(1) $n(2) 50Mb 10ms RED
$ns duplex-link $n(2) $n(3) 50Mb 10ms RED
$ns duplex-link $n(2) $n(4) 50Mb 10ms RED
$ns duplex-link $n(2) $n(6) 50Mb 10ms RED
$ns duplex-link $n(2) $n(7) 50Mb 10ms RED
$ns duplex-link $n(3) $n(4) 50Mb 10ms RED
$ns duplex-link $n(3) $n(6) 50Mb 10ms RED
$ns duplex-link $n(4) $n(5) 50Mb 10ms RED
$ns duplex-link $n(5) $n(6) 50Mb 10ms RED
$ns duplex-link $n(5) $n(9) 50Mb 10ms RED
$ns duplex-link $n(6) $n(7) 50Mb 10ms RED
$ns duplex-link $n(6) $n(8) 50Mb 10ms RED
$ns duplex-link $n(6) $n(9) 50Mb 10ms RED
$ns duplex-link $n(7) $n(8) 50Mb 10ms RED
$ns duplex-link $n(8) $n(9) 50Mb 10ms RED
$ns duplex-link $n(9) $n(10) 50Mb 10ms RED
$ns duplex-link $n(9) $n(11) 50Mb 10ms RED

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
```

```
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down


#Define a 'finish' procedure
proc finish {} {
        global ns nf tf f1 f2
        #Close the output files
        $ns flush-trace
        close $nf
        close $tf
        close $f1
        close $f2

        #Call xgraph to display the results
        exec nam out.nam &
        exec xgraph perdida-cbr perdida-exp -geometry 800x400 &

        exit 0
}

proc varia_CBR {} {
      global trafico1 tasa1

        #Get an instance of the simulator
        set ns [Simulator instance]
        #Set the time after which the procedure should be called again

      set tasa1 [expr $tasa1+2000000]

   # Va incrementando la tasa hasta 40Mbps
   if {$tasa1<=40000000} {
        $trafico1 set rate_ $tasa1
      set time 0.1
      set now [$ns now]
      $ns at [expr $now+$time] "varia_CBR"
   }

}
```

```
proc record {} {
        global final1 final1e f1 f2
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final1e set nlost_]
           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"
           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final1e set nlost_ 0
           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}


# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1
$ns attach-agent $n(10) $final1

set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen1e [new Agent/UDP]
set final1e [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1e
$ns attach-agent $n(10) $final1e

set trafico1e [new Application/Traffic/Exponential]
$trafico1e set packetSize_ 500
$trafico1e set rate_ 10000000
$trafico1e set burst_time_ .5
$trafico1e set iddle_time_ .1
$trafico1e attach-agent $origen1e

$ns connect $origen1e $final1e
```

```
# Trafico desde el nodo 1 al nodo 11  FTP
set origen2 [new Agent/TCP/FullTcp]
set final2 [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2
$ns attach-agent $n(11) $final2

# set up TCP-level connections
$final2 listen
$origen2 set window_ 100

set trafico2 [new Application/FTP]
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Trafico desde el nodo 1 al nodo 11  TELNET
set origen2t [new Agent/TCP/FullTcp]
set final2t [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2t
$ns attach-agent $n(11) $final2t

# set up TCP-level connections
$final2t listen
$origen2t set window_ 100

set trafico2t [new Application/Telnet]
$trafico2t set interval_ .1
$trafico2t attach-agent $origen2t

$ns connect $origen2t $final2t


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen1e set class_ 2
$ns color 2 Blue
$origen2 set class_ 3
$ns color 3 Green
$origen2t set class_ 4
$ns color 4 Brown


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico1e start"
$ns at 0.5 "$trafico2 start"
$ns at 0.5 "$trafico2t start"
```

```
$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico1e stop"
$ns at 6 "$trafico2 stop"
$ns at 6 "$trafico2t stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación de un planificador de tipo FQ

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]

$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}

#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms FQ
$ns duplex-link $n(1) $n(2) 50Mb 10ms FQ
$ns duplex-link $n(2) $n(3) 50Mb 10ms FQ
$ns duplex-link $n(2) $n(4) 50Mb 10ms FQ
$ns duplex-link $n(2) $n(6) 50Mb 10ms FQ
$ns duplex-link $n(2) $n(7) 50Mb 10ms FQ
$ns duplex-link $n(3) $n(4) 50Mb 10ms FQ
$ns duplex-link $n(3) $n(6) 50Mb 10ms FQ
$ns duplex-link $n(4) $n(5) 50Mb 10ms FQ
$ns duplex-link $n(5) $n(6) 50Mb 10ms FQ
$ns duplex-link $n(5) $n(9) 50Mb 10ms FQ
$ns duplex-link $n(6) $n(7) 50Mb 10ms FQ
$ns duplex-link $n(6) $n(8) 50Mb 10ms FQ
$ns duplex-link $n(6) $n(9) 50Mb 10ms FQ
$ns duplex-link $n(7) $n(8) 50Mb 10ms FQ
$ns duplex-link $n(8) $n(9) 50Mb 10ms FQ
$ns duplex-link $n(9) $n(10) 50Mb 10ms FQ
$ns duplex-link $n(9) $n(11) 50Mb 10ms FQ

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
```

```
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down


#Define a 'finish' procedure
proc finish {} {
    global ns nf tf f1 f2
#Close the output files
    $ns flush-trace
    close $nf
    close $tf
    close $f1
    close $f2

#Call xgraph to display the results
   exec nam out.nam &
   exec xgraph perdida-cbr perdida-exp perdida-ftp perdida-telnet -geometry 800x400 &

   exit 0
}

proc varia_CBR {} {
       global trafico1 tasa1

          #Get an instance of the simulator
          set ns [Simulator instance]
          #Set the time after which the procedure should be called again

       set tasa1 [expr $tasa1+2000000]

   # Va incrementando la tasa hasta 40Mbps
   if {$tasa1<=40000000} {
          $trafico1 set rate_ $tasa1
       set time 0.1
       set now [$ns now]
       $ns at [expr $now+$time] "varia_CBR"
   }

}
```

```
proc record {} {
        global final1 final1e f1 f2
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final1e set nlost_]

           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"

           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final1e set nlost_ 0

           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1
$ns attach-agent $n(10) $final1

set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen1e [new Agent/UDP]
set final1e [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1e
$ns attach-agent $n(10) $final1e

set trafico1e [new Application/Traffic/Exponential]
$trafico1e set packetSize_ 500
$trafico1e set rate_ 10000000
$trafico1e set burst_time_ .5
$trafico1e set iddle_time_ .1
$trafico1e attach-agent $origen1e
```

```
$ns connect $origen1e $final1e

# Trafico desde el nodo 1 al nodo 11  FTP
set origen2 [new Agent/TCP/FullTcp]
set final2 [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2
$ns attach-agent $n(11) $final2

# set up TCP-level connections
$final2 listen
$origen2 set window_ 100

set trafico2 [new Application/FTP]
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Trafico desde el nodo 1 al nodo 11  TELNET
set origen2t [new Agent/TCP/FullTcp]
set final2t [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2t
$ns attach-agent $n(11) $final2t

# set up TCP-level connections
$final2t listen
$origen2t set window_ 100

set trafico2t [new Application/Telnet]
$trafico2t set interval_ .1
$trafico2t attach-agent $origen2t

$ns connect $origen2t $final2t


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen1e set class_ 2
$ns color 2 Blue
$origen2 set class_ 3
$ns color 3 Green
$origen2t set class_ 4
$ns color 4 Brown


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico1e start"
```

```
$ns at 0.5 "$trafico2 start"
$ns at 0.5 "$trafico2t start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico1e stop"
$ns at 6 "$trafico2 stop"
$ns at 6 "$trafico2t stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación de un planificador de tipo WRR (primer escenario)

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]

$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}


#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 50Mb 10ms DropTail

$ns simplex-link $n(2) $n(6) 50Mb 10ms CBQ/WRR
$ns simplex-link $n(6) $n(2) 50Mb 10ms DropTail

$ns duplex-link $n(2) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(8) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(10) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 50Mb 10ms DropTail

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
```

```
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down

#---------------------------------------------------------------------
#Configure CBQ - WRR
#---------------------------------------------------------------------

# sintaxis: $cbqclass setparams parent okborrow allot maxidle prio level
# [[Manual 7.3]]
# parent=none indica que es el raiz
# okborrow booleano que indica que la clase puede recibir ancho de banda desde su padre
# allot fraccion de ancho de banda maximo (0.0 y 1.0). Este limite se
#       aumenta automaticamente si okborrow esta a true y no se esta aprovechando todo
#       el ancho de banda del enlace por otros traficos.
#       Si esta a false se desperdiciaria ancho de banda.
# maxidle maxima cantidad de tiempo que pueden estar paquetes en espera

# prio prioridad (0 a 10; 0 max prioridad)
# level las hojas del arbol son siempre nivel 1; padres de las hojas son 2...
# extradelay incremento del retardo


# Segun estos valores perdera trafico Expo pues es el menos prioritario
# Se pueden cambiar las colas Queue/RED por colas Queue/DropTail para ver si
# las graficas  son distintas.


set cbqlink [$ns link $n(2) $n(6)]
set topclass [new CBQClass]
$topclass setparams none 0 1 auto 8 2 0

#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
```

```
$class1 setparams $topclass true 0.7 auto 7 1 0

#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass true 0.3 auto 8 1 0

#FTP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.1 auto 6 1 0

#Telnet
set class4 [new CBQClass]
set queue4 [new Queue/RED]
$class4 install-queue $queue4
$class4 setparams $topclass true 0.1 auto 5 1 0

$cbqlink insert $topclass
$cbqlink insert $class1
$cbqlink insert $class2
$cbqlink insert $class3
$cbqlink insert $class4

$cbqlink bind $class1 1 ; # fid 1
$cbqlink bind $class2 2 ; # fid 2
$cbqlink bind $class3 3 ; # fid 3
$cbqlink bind $class4 4 ; # fid 4
#----------------



#Define a 'finish' procedure
proc finish {} {
        global ns nf tf f1 f2
        #Close the output files
        $ns flush-trace
        close $nf
        close $tf
        close $f1
        close $f2
        #Call xgraph to display the results
        exec nam out.nam &
        exec xgraph perdida-cbr perdida-exp -geometry 800x400 &

        exit 0
}
```

```
proc varia_CBR {} {
        global trafico1 tasa1

           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again

        set tasa1 [expr $tasa1+2000000]

    # Va incrementando la tasa hasta 40Mbps
    if {$tasa1<=40000000} {
           $trafico1 set rate_ $tasa1
        set time 0.1
        set now [$ns now]
        $ns at [expr $now+$time] "varia_CBR"
    }

}

proc record {} {
        global final1 final1e f1 f2
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final1e set nlost_]
           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"
           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final1e set nlost_ 0
           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1
$ns attach-agent $n(10) $final1

######################
$origen1 set fid_ 1
```

```
set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen1e [new Agent/UDP]
set final1e [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1e
$ns attach-agent $n(10) $final1e

#####################
$origen1e set fid_ 2

set trafico1e [new Application/Traffic/Exponential]
$trafico1e set packetSize_ 500
$trafico1e set rate_ 10000000
$trafico1e set burst_time_ .5
$trafico1e set iddle_time_ .1
$trafico1e attach-agent $origen1e

$ns connect $origen1e $final1e

# Trafico desde el nodo 1 al nodo 11  FTP
set origen2 [new Agent/TCP/FullTcp]
set final2 [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2
$ns attach-agent $n(11) $final2

#####################
$origen2 set fid_ 3

# set up TCP-level connections
$final2 listen
$origen2 set window_ 100

set trafico2 [new Application/FTP]
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Trafico desde el nodo 1 al nodo 11  TELNET
set origen2t [new Agent/TCP/FullTcp]
set final2t [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2t
$ns attach-agent $n(11) $final2t
```

```
####################
$origen2t set fid_ 4

# set up TCP-level connections
$final2t listen
$origen2t set window_ 100

set trafico2t [new Application/Telnet]
$trafico2t set interval_ .1
$trafico2t attach-agent $origen2t

$ns connect $origen2t $final2t


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen1e set class_ 2
$ns color 2 Blue
$origen2 set class_ 3
$ns color 3 Green
$origen2t set class_ 4
$ns color 4 Brown


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico1e start"
$ns at 0.5 "$trafico2 start"
$ns at 0.5 "$trafico2t start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico1e stop"
$ns at 6 "$trafico2 stop"
$ns at 6 "$trafico2t stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación de un planificador de tipo WRR (2º escenario)

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]

$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}


#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 50Mb 10ms DropTail

$ns simplex-link $n(2) $n(6) 50Mb 10ms CBQ/WRR
$ns simplex-link $n(6) $n(2) 50Mb 10ms DropTail

$ns duplex-link $n(2) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(8) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(10) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 50Mb 10ms DropTail

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right-up
```

```
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down

#---------------------------------------------------------------------
#Configure CBQ - WRR
#---------------------------------------------------------------------

# sintaxis: $cbqclass setparams parent okborrow allot maxidle prio level
# [[Manual 7.3]]
# parent=none indica que es el raiz
# okborrow booleano que indica que la clase puede recibir ancho de banda desde su padre
# allot fraccion de ancho de banda maximo (0.0 y 1.0). Este limite se
#       aumenta automaticamente si okborrow esta a true y no se esta aprovechando todo
#       el ancho de banda del enlace por otros traficos.
#       Si esta a false se desperdiciaria ancho de banda.
# maxidle maxima cantidad de tiempo que pueden estar paquetes en espera

# prio prioridad (0 a 10; 0 max prioridad)
# level las hojas del arbol son siempre nivel 1; padres de las hojas son 2...
# extradelay incremento del retardo


# Segun estos valores perdera trafico Expo pues es el menos prioritario
# Se pueden cambiar las colas Queue/RED por colas Queue/DropTail para ver si
# las graficas  son distintas.


set cbqlink [$ns link $n(2) $n(6)]
set topclass [new CBQClass]
$topclass setparams none 0 1 auto 8 2 0

#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
$class1 setparams $topclass true .7 auto 7 1 0
```

```
#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass false .15 auto 7 1 0

#FTP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.1 auto 5 1 0

#Telnet
set class4 [new CBQClass]
set queue4 [new Queue/RED]
$class4 install-queue $queue4
$class4 setparams $topclass true 0.1 auto 5 1 0

$cbqlink insert $topclass
$cbqlink insert $class1
$cbqlink insert $class2
$cbqlink insert $class3
$cbqlink insert $class4

$cbqlink bind $class1 1 ; # fid 1
$cbqlink bind $class2 2 ; # fid 2
$cbqlink bind $class3 3 ; # fid 3
$cbqlink bind $class4 4 ; # fid 4
#----------------



#Define a 'finish' procedure
proc finish {} {
        global ns nf tf f1 f2
        #Close the output files
        $ns flush-trace
        close $nf
        close $tf
        close $f1
        close $f2

        #Call xgraph to display the results
        exec nam out.nam &
        exec xgraph perdida-cbr perdida-exp -geometry 800x400 &

        exit 0
}
```

```
proc varia_CBR {} {
        global trafico1 tasa1

           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again

        set tasa1 [expr $tasa1+2000000]

     # Va incrementando la tasa hasta 40Mbps
     if {$tasa1<=40000000} {
           $trafico1 set rate_ $tasa1
        set time 0.1
        set now [$ns now]
        $ns at [expr $now+$time] "varia_CBR"
     }

}

proc record {} {
        global final1 final1e f1 f2
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final1e set nlost_]
           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"
           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final1e set nlost_ 0
           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1
$ns attach-agent $n(10) $final1

#####################
$origen1 set fid_ 1

set trafico1 [new Application/Traffic/CBR]
```

```
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen1e [new Agent/UDP]
set final1e [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen1e
$ns attach-agent $n(10) $final1e

######################
$origen1e set fid_ 2

set trafico1e [new Application/Traffic/Exponential]
$trafico1e set packetSize_ 500
$trafico1e set rate_ 10000000
$trafico1e set burst_time_ .5
$trafico1e set iddle_time_ .1
$trafico1e attach-agent $origen1e

$ns connect $origen1e $final1e

# Trafico desde el nodo 1 al nodo 11  FTP
set origen2 [new Agent/TCP/FullTcp]
set final2 [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2
$ns attach-agent $n(11) $final2

######################
$origen2 set fid_ 3

# set up TCP-level connections
$final2 listen
$origen2 set window_ 100

set trafico2 [new Application/FTP]
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Trafico desde el nodo 1 al nodo 11  TELNET
set origen2t [new Agent/TCP/FullTcp]
set final2t [new Agent/TCP/FullTcp]
$ns attach-agent $n(1) $origen2t
$ns attach-agent $n(11) $final2t

######################
```

```
$origen2t set fid_ 4

# set up TCP-level connections
$final2t listen
$origen2t set window_ 100

set trafico2t [new Application/Telnet]
$trafico2t set interval_ .1
$trafico2t attach-agent $origen2t

$ns connect $origen2t $final2t


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen1e set class_ 2
$ns color 2 Blue
$origen2 set class_ 3
$ns color 3 Green
$origen2t set class_ 4
$ns color 4 Brown


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico1e start"
$ns at 0.5 "$trafico2 start"
$ns at 0.5 "$trafico2t start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico1e stop"
$ns at 6 "$trafico2 stop"
$ns at 6 "$trafico2t stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación del tráfico generado por una aplicación multimedia (primer escenario)

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]
set f5 [open perdida-udp w]

set f6 [open rend-cbr w]
set f7 [open rend-exp w]
set f10 [open rend-udp w]
$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}

#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms RED
$ns duplex-link $n(1) $n(2) 50Mb 10ms RED
$ns duplex-link $n(2) $n(3) 50Mb 10ms RED
$ns duplex-link $n(2) $n(4) 50Mb 10ms RED
$ns duplex-link $n(2) $n(6) 50Mb 10ms RED
$ns duplex-link $n(2) $n(7) 50Mb 10ms RED
$ns duplex-link $n(3) $n(4) 50Mb 10ms RED
$ns duplex-link $n(3) $n(6) 50Mb 10ms RED
$ns duplex-link $n(4) $n(5) 50Mb 10ms RED
$ns duplex-link $n(5) $n(6) 50Mb 10ms RED
$ns duplex-link $n(5) $n(9) 50Mb 10ms RED
$ns duplex-link $n(6) $n(7) 50Mb 10ms RED
$ns duplex-link $n(6) $n(8) 50Mb 10ms RED
$ns duplex-link $n(6) $n(9) 50Mb 10ms RED
$ns duplex-link $n(7) $n(8) 50Mb 10ms RED
$ns duplex-link $n(8) $n(9) 50Mb 10ms RED
$ns duplex-link $n(9) $n(10) 50Mb 10ms RED
$ns duplex-link $n(9) $n(11) 50Mb 10ms RED

# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
```

```
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down


#Define a 'finish' procedure
proc finish {} {
        global ns nf tf f1 f2 f5 f6 f7  f10
        #Close the output files
        $ns flush-trace
        close $nf
        close $tf
        close $f1
        close $f2
        close $f5

        close $f6
        close $f7
        close $f10

        #Call xgraph to display the results

        exec nam out.nam &

        exec xgraph perdida-cbr perdida-exp perdida-udp -geometry 800x400 &
        exec xgraph rend-cbr rend-exp rend-udp -geometry 800x400 &
        exit 0
}

proc varia_CBR {} {
      global trafico1 tasa1

        #Get an instance of the simulator
        set ns [Simulator instance]
        #Set the time after which the procedure should be called again

      set tasa1 [expr $tasa1+2000000]
```

```
   # Va incrementando la tasa hasta 45Mbps
   if {$tasa1<=45000000} {
         $trafico1 set rate_ $tasa1
      set time 0.1
      set now [$ns now]
      $ns at [expr $now+$time] "varia_CBR"
   }

}

proc record {} {
      global final1 final2 final3 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10
         #Get an instance of the simulator
         set ns [Simulator instance]
         #Set the time after which the procedure should be called again
      set time 0.1
      set pkd1 [$final1 set nlost_]
      set pkd2 [$final2 set nlost_]
      set pkd5 [$final3 set nlost_]

      set bw6 [$final1 set bytes_]
      set bw7 [$final2 set bytes_]
      set bw10 [$final3 set bytes_]


         #Get the current time
      set now [$ns now]
      puts $f1 "$now $pkd1"
      puts $f2 "$now $pkd2"
      puts $f5 "$now $pkd5"

      puts $f6 "$now [expr $bw6/$time*8/1000000]"
      puts $f7 "$now [expr $bw7/$time*8/1000000]"
      puts $f10 "$now [expr $bw10/$time*8/1000000]"

         #Reset the nlost_ values on the traffic sinks
      $final1 set nlost_ 0
      $final2 set nlost_ 0
      $final3 set nlost_ 0

      $final1 set bytes_ 0
      $final2 set bytes_ 0
      $final3 set bytes_ 0

         #Re-schedule the procedure
      $ns at [expr $now+$time] "record"
}

##############################
```

```
# TRAZA STAR WARS

set tfile [new Tracefile]
$tfile filename starwars.nsformat

set origen3 [new Agent/UDP]
set final3 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen3
$ns attach-agent $n(10) $final3

set trafico3 [new Application/Traffic/Trace]
$trafico3 attach-tracefile $tfile
$trafico3 attach-agent $origen3
$ns connect $origen3 $final3


# Trafico desde el nodo 1 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(1) $origen1
$ns attach-agent $n(10) $final1

set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen2 [new Agent/UDP]
set final2 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen2
$ns attach-agent $n(10) $final2

set trafico2 [new Application/Traffic/Exponential]
$trafico2 set packetSize_ 500
$trafico2 set rate_ 15000000
$trafico2 set burst_time_ .3
$trafico2 set iddle_time_ .2
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
```

```
$origen2 set class_ 2
$ns color 2 Blue
$origen3 set class_ 5
$ns color 5 Yellow


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico2 start"

$ns at 0.5 "$trafico3 start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico2 stop"

$ns at 6 "$trafico3 stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Simulación del tráfico generado por una aplicación multimedia (2º escenario)

```
#Create a simulator object
set ns [new Simulator]

## $ns rtproto DV

#Open the output files
set tf [open out.tr w]
set f1 [open perdida-cbr w]
set f2 [open perdida-exp w]
set f5 [open perdida-udp w]

set f6 [open rend-cbr w]
set f7 [open rend-exp w]
set f10 [open rend-udp w]
$ns trace-all $tf

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Crear los 12 nodos
for {set i 0} {$i<=11} {incr i} {
   set n($i) [$ns node]
}

#Connect the nodes
$ns duplex-link $n(0) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 50Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 50Mb 10ms DropTail

$ns simplex-link $n(2) $n(6) 50Mb 10ms CBQ/WRR
$ns simplex-link $n(6) $n(2) 50Mb 10ms DropTail

$ns duplex-link $n(2) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 50Mb 10ms DropTail
$ns duplex-link $n(3) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 50Mb 10ms DropTail
$ns duplex-link $n(5) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 50Mb 10ms DropTail
$ns duplex-link $n(8) $n(9) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(10) 50Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 50Mb 10ms DropTail
```

```
# Orientacion grafica para nam
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-up
$ns duplex-link-op $n(2) $n(6) orient right
$ns duplex-link-op $n(2) $n(7) orient right-down
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(6) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(5) $n(6) orient left-down
$ns duplex-link-op $n(5) $n(9) orient right-down
$ns duplex-link-op $n(6) $n(7) orient left-down
$ns duplex-link-op $n(6) $n(8) orient right-down
$ns duplex-link-op $n(6) $n(9) orient right
$ns duplex-link-op $n(7) $n(8) orient right
$ns duplex-link-op $n(8) $n(9) orient right-up
$ns duplex-link-op $n(9) $n(10) orient right-up
$ns duplex-link-op $n(9) $n(11) orient right-down


#---------------------------------------------------------------------
#Configure CBQ - WRR
#---------------------------------------------------------------------

# sintaxis: $cbqclass setparams parent okborrow allot maxidle prio level
# [[Manual 7.3]]
# parent=none indica que es el raiz
# okborrow booleano que indica que la clase puede recibir ancho de banda desde su padre
# allot fraccion de ancho de banda maximo (0.0 y 1.0). Este limite se
#       aumenta automaticamente si okborrow esta a true y no se esta aprovechando todo
#       el ancho de banda del enlace por otros traficos.
#       Si esta a false se desperdiciaria ancho de banda.
# maxidle maxima cantidad de tiempo que pueden estar paquetes en espera

# prio prioridad (0 a 10; 0 max prioridad)
# level las hojas del arbol son siempre nivel 1; padres de las hojas son 2...
# extradelay incremento del retardo


# Segun estos valores perdera trafico Expo pues es el menos prioritario
# Se pueden cambiar las colas Queue/RED por colas Queue/DropTail para ver si
# las graficas  son distintas.


set cbqlink [$ns link $n(2) $n(6)]
set topclass [new CBQClass]
$topclass setparams none 0 1 auto 8 2 0
```

```
#CBR
set class1 [new CBQClass]
set queue1 [new Queue/RED]
$class1 install-queue $queue1
$class1 setparams $topclass true 0.8 auto 7 1 0

#Expo
set class2 [new CBQClass]
set queue2 [new Queue/RED]
$class2 install-queue $queue2
$class2 setparams $topclass true 0.3 auto 7 1 0

#UDP
set class3 [new CBQClass]
set queue3 [new Queue/RED]
$class3 install-queue $queue3
$class3 setparams $topclass true 0.2 auto 5 1 0

$cbqlink insert $topclass
$cbqlink insert $class1
$cbqlink insert $class2
$cbqlink insert $class3

$cbqlink bind $class1 1 ; # fid 1
$cbqlink bind $class2 2 ; # fid 2
$cbqlink bind $class3 3 ; # fid 3
#----------------




#Define a 'finish' procedure
proc finish {} {
        global ns nf tf f1 f2 f5 f6 f7 f10
        #Close the output files
        $ns flush-trace
        close $nf
        close $tf
        close $f1
        close $f2
        close $f5
        close $f6
        close $f7
        close $f10
        #Call xgraph to display the results
        exec nam out.nam &
        exec xgraph perdida-cbr perdida-exp perdida-udp -geometry 800x400 &
        exec xgraph rend-cbr rend-exp rend-udp -geometry 800x400 &
        exit 0
}
```

```
proc varia_CBR {} {
        global trafico1 tasa1

           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again

        set tasa1 [expr $tasa1+2000000]

     # Va incrementando la tasa hasta 45Mbps
     if {$tasa1<=45000000} {
           $trafico1 set rate_ $tasa1
        set time 0.1
        set now [$ns now]
        $ns at [expr $now+$time] "varia_CBR"
     }

}

proc record {} {
        global final1 final2 final3 f1 f2 f5 f6 f7 f10
           #Get an instance of the simulator
           set ns [Simulator instance]
           #Set the time after which the procedure should be called again
        set time 0.1
        set pkd1 [$final1 set nlost_]
        set pkd2 [$final2 set nlost_]
        set pkd5 [$final3 set nlost_]

        set bw6 [$final1 set bytes_]
        set bw7 [$final2 set bytes_]
        set bw10 [$final3 set bytes_]


           #Get the current time
        set now [$ns now]
        puts $f1 "$now $pkd1"
        puts $f2 "$now $pkd2"
        puts $f5 "$now $pkd5"

        puts $f6 "$now [expr $bw6/$time*8/1000000]"
        puts $f7 "$now [expr $bw7/$time*8/1000000]"
        puts $f10 "$now [expr $bw10/$time*8/1000000]"

           #Reset the nlost_ values on the traffic sinks
        $final1 set nlost_ 0
        $final2 set nlost_ 0
        $final3 set nlost_ 0
```

```
        $final1 set bytes_ 0
        $final2 set bytes_ 0
        $final3 set bytes_ 0

           #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

##############################
# TRAZA STAR WARS

set tfile [new Tracefile]
$tfile filename starwars.nsformat

set origen3 [new Agent/UDP]
set final3 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen3
$ns attach-agent $n(10) $final3

$origen3 set fid_ 3

set trafico3 [new Application/Traffic/Trace]
$trafico3 attach-tracefile $tfile
$trafico3 attach-agent $origen3
$ns connect $origen3 $final3

# Trafico desde el nodo 0 al nodo 10   CBR
set tasa1 1000000

set origen1 [new Agent/UDP]
set final1 [new Agent/LossMonitor]
$ns attach-agent $n(1) $origen1
$ns attach-agent $n(10) $final1

$origen1 set fid_ 1

set trafico1 [new Application/Traffic/CBR]
$trafico1 set packetSize_ 2000
$trafico1 set rate_ $tasa1   # Tasa inicial
$trafico1 attach-agent $origen1
$ns connect $origen1 $final1

# Trafico desde el nodo 0 al nodo 10   Exponencial

set origen2 [new Agent/UDP]
set final2 [new Agent/LossMonitor]
$ns attach-agent $n(0) $origen2
$ns attach-agent $n(10) $final2

$origen2 set fid_ 2
```

```
set trafico2 [new Application/Traffic/Exponential]
$trafico2 set packetSize_ 500
$trafico2 set rate_ 15000000
$trafico2 set burst_time_ .3
$trafico2 set iddle_time_ .2
$trafico2 attach-agent $origen2

$ns connect $origen2 $final2


# Colores para nam
$origen1 set class_ 1
$ns color 1 Red
$origen2 set class_ 2
$ns color 2 Blue
$origen3 set class_ 3
$ns color 3 Yellow


$ns at 0.0 "record"
#Start the traffic sources
$ns at 0.5 "$trafico1 start"
$ns at 0.5 "$trafico2 start"

$ns at 0.5 "$trafico3 start"

$ns at 1.0 "varia_CBR"

##$ns rtmodel-at 1.0 down $n(6) $n(9)

#Stop the traffic sources
$ns at 6 "$trafico1 stop"
$ns at 6 "$trafico2 stop"

$ns at 6 "$trafico3 stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 6.5 "finish"

#Run the simulation
$ns run
```

Bibliografía

- B. Carpenter. Architectural Principles of the Internet, RFC 1958, IETF, www.ietf.org, Junio 1996. rfc1958.txt.
- M. Allman, A. Folk. On the Effective Evaluation of TCP. ACM Computer Communication Review. Octubre 1999.
- S. Vegesna: IP Quality of Service (Cisco Networking Fundamentals). Cisco Press. 2001.
- G. Armitage: Quality of Service in IP Networks. Que, 1era edición. 2000.
- The Internet Engineering Task Forc, http://www.ietf.org
- ReSerVation Protocol-RSVP, http://www.isi.edu/rsvp/.
- Differentiated Services (Diffserv), http://www.ietf.org/html.charters/diffserv-charter.html.
- Multiprotocol Label Switching (MPLS), http://www.ietf.org/html.charters/mpls-charter.html.
- "Why We Don't Know How To Simulate The Internet", http://www.aciri.ora/floyd/papers/wsc.ps.
- Tutorial del ns en http://www.isi.edu/nsnam/ns/tutorial/index.html.
- NS by Example en http://nile.wpi.edu/NS/.
- La gran mayoría de las contribuciones del ns se encuentran en http://www.isi.edu/nsnam/ns/ns-contributed.html.
- L. Bres1au, S. Jamin, S. Shenker, "Comments on the Performance of Measurement-Based Admission Control Algorithms" Proc. of IEEE Infocom 2000, March 26-30, 2000, http://irl.eecs.umich.edu/Jamin/papers/mbac/infocom00.ps.
- Y. Lai, S. Tsai, "Unfairness of Measurement-based Admission Controls in a Heterogeneous Environment", Parallel and Distributed Systems, 2001. ICPADS 2001. Proceeding. IEEE.

- C. Semeria, “Supporting Differentiated Service Classes: Queue Scheduling Disciplines”, White Papers, Juniper Network. 2001.

- R. Guérin, V. Peris, “Quality-of-Service in packet networks: basic mechanisms and directions”, Computers Networks, Vol No 31, 1999.

- C. Semeria, “Supporting Differentiated Service Classes: Active Queue Memory Management”, White Papers, Juniper Network. 2002.

Autorización

Los abajo firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado para este proyecto.

Fdo.: Saulo Barajas Fernandez

Fdo.: Santiago Martínez Sanz

Fdo.: Jaime Parodi Bardón